CEEMAS'99

PROCEEDINGS of the 1st International Workshop of Central and Eastern Europe on Multi-Agent Systems CEEMAS'99

June 1–4, 1999
St.Petersburg, Russia

Russian Foundation of Basic Research

European Office of Aerospace Research and Development (EOARD)

European Office of Naval Research Europe (ONREUR)

AgentLink: Network of Excellence for Agent-based Computing

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 29 June 1999	3. REPORT TYPE AND DATES COVERED Conference Proceedings

4. TITLE AND SUBTITLE The First International Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS'99)	5. FUNDING NUMBERS F61775-99-WF005
6. AUTHOR(S) Conference Committee	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) St.Petersburg Institute For Informatics & Automation of the Russian Academy of Sciences 39, 14th Liniya St. Peterburg 199178 Russia	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD PSC 802 BOX 14 FPO 09499-0200	10. SPONSORING/MONITORING AGENCY REPORT NUMBER CSP 99-5005

11. SUPPLEMENTARY NOTES

Consists of three volumes: Agenda, Program, and Proceedings

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (Maximum 200 words)

The Final Proceedings for The First International Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS'99), 30 May 1999 - 3 June 1999

This is an interdisciplinary conference. Topics include 1. Agent-based modeling for wide range of combinatorial problems solving, 2. Negotiation, cooperation and conflict resolution in agent-based systems 3. Multi-agent systems and knowledge discovery: agent-based models of data mining and knowledge discovery 4.models of uncertainty applicable to development of agents and agent-based techniques to deal with uncertainty.

14. SUBJECT TERMS EOARD, Agent Based Systems, data mining, knowledge bases	15. NUMBER OF PAGES 361
	16. PRICE CODE N/A

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19, SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

PROCEEDINGS

# of the 1st International Workshop of Central and Eastern Europe on Multi-Agent Systems CEEMAS'99

June 1-4, 1999
St. Petersburg, Russia

ORGANIZED BY:

Department of Informatics, Computer Sciences and Automation of the Russian Academy of Sciences

Russian Association of Artificial Intelligence

St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS)

Department of Computer Science, Technical University of Mining and Metallurgy of Cracow, Poland

SPONSORED BY:

Russian Foundation of Basic Research

European Office of Aerospace Research and Development (EOARD)

European Office of Naval Research Europe (ONREUR)

AgentLink: Network of Excellence for Agent-based Computing

CEEMAS'99 logo and cover design by Dmitry Kirsanov Studio, **www.kirsanov.com**

Proceedings of the First International Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS`99). May 31 – June 04, 1999. St.-Petersburg, Russia.
ISBN 5 – 7452 – 0029 – 4
Труды Первого Международного семинара Центральной и Восточной Европы по многоагентным системам. 31 мая – 04 июня, 1999 г. Санкт-Петербург, Россия.
ISBN 5 – 7452 – 0029 – 4

275 лет
Российская академия наук
Основана в 1724 году

CONTENTS

Program Committee

Co-Chairs:

K.Cetnarowicz (Technical University of Mining and Metallurgy (AGH) of Cracow, Poland)
Y.Demazeau (Lab. LEIBNIZ-Institut IMAG, France)
V.Gorodetski (SPIIRAS, Russia)

Program Committee members:

S.Ambroszkiewicz (Institute of Computer Science, Poland)
P.Barnev (Institute of Mathematics and Informatics, Bulgaria)
M.Boman (DSVSU/KTH,Sweden)
J.Debenham (University of Technology, Sydney, Australia)
B.Dunin-Keplicz (Institute of Informatics, Warsaw University, Poland)
A.Florea (University of Bucharest, Romania)
M.Ivanovic (University of Novi Sad, Yugoslavia)
N.Jennings (Queen Mary and Westfield College, University of London, UK)
E.Jones (USAF Research Lab., USA)
Yu.Karpov (St. Petersburg State Technical University, Russia)
W.Kloeśgen (GMD, Germany)
B.McKinney (USAF/EOARD, USA)
E.Nawarecki (Technical University of Mining and Metallurgy (AGH) of Cracow, Poland)
G.Osipov (Programme Systems Institute, Pereslavl-Zalessky, Russia)
I.Plander (Bratislava, Slovakia)
Z.Z.Shi (Institute of Computer Technology, China)
J.S.Sichman (University of San Paulo, Brazil)
A.Slissenko (University Paris-12, France)
A.Smirnov (SPIIRAS, St. Petersburg, Russia)
V.Srovnal (Technical University of Ostrava, Czech Republic)
V.Stefanuk (Institute for Problems of Information Transmission, Russia)
V.Tarassov (Bauman Moscow State Technical University, Russia)
J.Vancza (Computer and Automation Research Institute, Hungary)

Program Committee Scientific Secretary

Alexandre L. Touloupiev (SPIIRAS, St. Petersburg, Russia)
(e-mail: ceemas@mail.iias.spb.su)

Program Committee Address

SPIIRAS, 39, 14th Line V.O.
199178, St. Petersburg
Russia

tel. +7(812)328-54-11
fax +7(812)328-06-85
e-mail ceemas@mail.iias.spb.su
http://space.iias.spb.su/ai/english/ceemas'99.htm

Organizing Committee

General Chair:
R. Yusupov (SPIIRAS, St. Petersburg)

Organizing Committee Members:

I. Podnozova (SPIIRAS, St. Petersburg)
A. Tkatch (SPIIRAS, St. Petersburg)
V. Nesterov (SPIIRAS, St. Petersburg)
L. Fedorchenko (SPIIRAS, St. Petersburg)
A. Pasmurov (RESTEK, St. Petersburg)

Organizing Committee Secretary

Irina A. Tsygankova (SPIIRAS, St. Petersburg)
(e-mail: yralts@mail.iias.spb.su)

Organizing Committee Address:
SPIIRAS, 39,14th Line V.O.
St. Petersburg, 199178
Russia

Fax: +7(812) 328 06 85
Phone: +7(812) 328 54 11
E-mail: yralts@mail.iias.spb.su
(Irina A. Tsygankova)
http://space.iias.spb.su/ai/english/ceemas'99.htm

Reviewers

S. Ambroszkiewicz (Poland)
Guilherme Bittencourt (Brazil)
Magnus Boman (Sweden)
Hu Cao (PR China)
Barbara Dunin-Keplicz (Poland)
Krzysztof Cetnarowicz (Poland)
John Debenham (Australia)
Yves Demazeau (France)
Adina Magda Florea (Romania)
V.Gorodetski (Russia)
Mirjana Ivanovic (Yugoslavia)
Nick Jennings (United Kingdom)
Andrea Julenyova (Slovakia)
Yuri G. Karpov (Russia)
W. Kloesgen (Germany)
Edward Nawarecki (Poland)
Stefan Neuschl (Slovakia)
Gennady Osipov (Russia)
Leonardo Azevedo Scardua (Brazil)
Zhongzhi Shi (PR China)
Jaime S. Sichman (Brazil)
Anatol Slissenko (France)
Alexander Smirnov (Russia)
Vilem Srovnal (Czech Republic)
Vadim Stefanuk (Russia)
Jaromir Suchanek (Slovakia)
V.B. Tarassov (Russia)
J. Vancza (Hungary)
Jiao Wenping (PR China)

FOREWORD

Intelligent agents and agent-based technology form a new paradigm for developing complex and large scale software applications. Currently Intelligent agents and Multi-agent systems are being used in an increasingly wide variety of applications, ranging from comparatively small systems like intelligent personal user assistants, say, for e-mail filtering to large open complex systems of real time planning, scheduling and control such as air traffic control, shipping and other.

Multi-agent systems (MAS) and agent-based technologies are the focus of intense interest of many sub-fields in computer science and artificial intelligence, and in many aspects are based on their theoretic background. In addition MAS set their own theoretical challenges, particularly, in the areas of collective and cooperative behavior, dealing with uncertainty, knowledge engineering and learning, programming, etc.

On the other hand, MAS research is forming a new viewpoint on the traditional problems of artificial intelligence and is enriching the set of paradigms, methods and algorithms of artificial intelligence via raising new problems, proposing new approaches , etc.

The Workshop is considered by organizers as a natural extension of the two previous ones held in Poland in 1995 (DIMAS'95) and in St. Petersburg in 1997 (DAIMAS'97). These events were a big success. For example, researchers from 12 countries participated in DAIMAS'97. For more information including electronic versions of all presentations at DAIMAS'97 in English and in Russian see http://space.iias.spb.su/ai/english/windex.htm. While attending ICMAS'98 in Paris, the organizers of both events decided to join their efforts, and to draw in additional national inputs from Bulgaria, Hungary, Romania, Czech, Slovakia, Yugoslavia to establish a biennial Central and Eastern Europe Workshop on Multi-Agent Systems that hopefully will grow into a major agent-oriented scientific event in Central and Eastern Europe. The current event is entitled The First International Workshop of Central-Eastern Europe on Multi-agent Systems (CEEMAS'99).

The agenda of the Workshop proposes invited talks of well known scientists in the area of Multi-agent Systems and thorough discussion of selected hot problems of Multi-agent theory and practice according to the following technical sessions:

Session 1. Negotiation, Cooperation and Conflict Resolution in Agent-based Combinatorial Problems Solving.

Session 2. Agent-based Modeling of Combinatorial Problems Solving.

Session 3. Agent-based Modeling of Combinatorial Problems: Real Time Planning, Scheduling and Resource Allocation.

Session 4. Agent-based Modeling of Combinatorial Problems Solving.

Session 5. Multi-agent Systems and Knowledge Discovery: Mutual Impact.

We hope that CEEMAS'99 materials published in this volume will be of interest for the specialists in the area of Multi-agent systems and will draw attention of the corresponding scientific community to the new areas of potential applications of Multi-agent technology, interconnections and mutual impact of Multi-agent systems and other areas of Artificial Intelligence.

Chairman of the Organizing
Committee of the CEEMAS'99

Professor R.Yusupov

Co-chairman of the Program
Committee of the CEEMAS'99

Professor V.Gorodetski

PART I
Invited Papers

OPERATIONAL AGENT: A CONCEPT OF MIDDLE AGENT ARCHITECTURE*

(extended abstract)

Stanisław Ambroszkiewicz and Krzysztof Cetnarowicz
Email: `sambrosz@ipipan.waw.pl`, `cetnar@uci.agh.edu.pl`,
WWW: `http://www.ipipan.waw.pl/mas/` *and* `http://galaxy.uci.agh.edu.pl/~cetnar/`

Abstract. *We present experiences gained during realization of our ongoing project on modeling Agent Virtual Organizations (AVOs) [7]. The experiences presented in the paper concern various agent architectures we have tried to apply for the agents that were supposed to form and reform virtual organizations.*

Key words: *Agent architectures, M-agent, operational agent.*

1 Software agent

Let us start with the concept of software agent.
From Peng [27]:
"There is no consensus on the definition of software agent or of agency. Some people go so far to suggest that any piece of software or object that can perform a specific task is an agent. However, the prevailing opinion is that a software agent may exhibit three important general characteristics: *autonomy, adaptation,* and *cooperation.*
By 'autonomy' we mean that agents have their own agenda of goals and exhibit goal-directed behavior. They are not simply reactive, but can be proactive and take initiatives, as they deem appropriate. In this sense, agent systems can be viewed as a generalization of the client-server model in that each agent can be both a client and a server and can provide and request services to and from others.
'Adaptation' implies that agents are capable of adapting to the environment, which includes other agents and human users, and can learn from the experience in order to improve themselves in a changing environment.
'Cooperation' and coordination between agents is probably the most important feature of MAS. Unlike those stand-alone agents, agents in MAS collaborate with each other to achieve common goals. In other words, these agents *share* information, knowledge, and tasks among themselves. The intelligence of MAS is not only reflected by the expertise of individual agents but first of all is exhibited by the emerged collective behavior beyond individual agents."
In order to be a bit more formally let us present the definition of multi-agent system proposed by Y. Demazeau [15]. According to that definition MAS is defined as

$$MAS = (A, I, O, E)$$

where: A is a set of agents. I is a set of agents' interactions (see Ribeiro & Demazeau [30]) like communications in ACL [17], CNP (Contract Net Protocol [32]), auctions [31]. O is a set of static organizations with fixed roles for the agents to undertake (see Ferber [16] for overview). E is a specific environment.
According to this definition of MAS, interactions and static organizations form an infrastructure (global facilities) for the agents to cooperate.
Our view is that these very interactions and organizations (that are actually necessary as an infrastructure) should be extended to virtual interactions and organizations. These very organizations are created locally by the agents themselves in situation where they are needed by the agents as means for cooperation.
The main motive of our work is that "Intelligence" emerges from behavior of virtual organizations of operational agents!

2 Review of existing agent architectures

In the sequel we are going to compare and evaluate existing agent architectures.

2.1 Reactive agent architecture: bottom-up approach

Reactive agent is defined by a stimulus - response function F, such that for given stimulus s from the envi-

*Supported by the State Committee for Scientific Research under the grant No. 8 T11C 008 12

ronment and current state c of the agent, it determines reaction = $F(s, c)$.
Reactive agents do not have goals and are not proactive.

2.2 Cognitive agent: top down approach

Cognitive (deliberative) agent has symbolic representation of the world. It has several additional mental attitudes like motivational and intentional ones. Its epistemic attitude contains mutual knowledge, i.e. knowledge about other agents knowledge [1]. On the basis of these attitudes the agent performs planning, reasoning, etc..
The architecture of BDI-agent (see Bratman [10], Rao & Georgeff [29]) is here the most famous example. The inspiration of Belief, Desire, and Intentions is borrowed from a psychological view of human cognition. These very mental attitudes (belief, desire, and intentions) are taken as primitive notions on which the reasoning and planning process is based.
This architecture goes back to the traditional mainstream of symbolic Artificial Intelligence.

2.3 The need for middle architecture: hybrid architecture?

There is a wide spectrum for potential agent architectures between the reactive one and deliberative one. In fact, reactive architecture is extremely simple whereas deliberative one is extremely complex.
Hybrid architectures concept tries to overcome the limitations of deliberative and reactive architectures just by combining them. Usually these architectures are composed of several layers. For example in J. Mueller's InterRap architecture [25] only neighboring layers interact. Each layer of an InterRap agent is responsible for different activity. The bottom *behavior-based* layer implements reactive behavior and procedural knowledge. The middle *plan-based* layer contains a mechanism for constructing individual plans for the agent given current knowledge about the world. The top *cooperation* layer has capability to generate collaborative plans using special cooperative knowledge.
Such kinds of architectures do not seem to overcome the complexity of deliberative architecture. The reason for that is that the hybrid architectures actually extend deliberative architecture by adding reactive component. Hence, for some stimuli a hybrid agent behaves reactively according to bottom part of the architecture, whereas the rest if its behavior is determined by cognitive part.
Hence, it seems that the concept of hybrid architecture does not propose any qualitative progress here!

3 "Cooperation" as a test for evaluation agent architectures

Cooperation is it the key concept in MAS [20] so it may serve as the best test for evaluation of agent architectures.

- BDI-architectures exhibit good performance in relatively simple worlds with a small number of agents. Cooperation techniques developed for this kind of architecture are mainly theoretical ones. They are based on collective knowledge, goals, intentions commitments, see the notion of team formation and reconfiguration Cohen et al. [13], Kinney et al. [22], Dunin-Keplicz et al. [21].
- Reactive agent architectures are good solutions for large not structured environment with large uniform agent societies. For such environments emergence of cooperation (organizational) patterns of agent societies can be observed, see Ferber [16].

It seems that there is a critical level of complexity of $E + I + A$ where none of the agent architectures described above is appropriate in the sense that the agents are inefficient or unable to realize their goals.
This very critical level can be clearly seen for environments where fulfilling single agent's goal needs a complex workflow to be performed and the workflow involves a large number of self interested agent to cooperate with each other.
What BDI-agents can do in this case?
-they compute joint plans, goals, intentions;
- they try to make (negotiate with) other agents to commit to them;
- all this takes time, so that after achieving this, the plans, goals, intentions are out-of-date!
What reactive agents can do?
- not too much, they are to simple!

4 Our proposal

The criticism presented in the previous section concerns reactive and cognitive agent architectures not in general but only for applications in complex environments. Of course there are environments for which these architectures are appropriate. Our experiences (from modeling cooperation mechanisms in general and agent virtual organizations in particular [6, 7]) clearly show that for complex environments neither bottom-up nor top-down approach is appropriate. For such complex environments a new middle approach is needed. The following architecture of M-agent may be seen as a proposal towards this direction.

4.1 M-agent as a candidate for middle agent

The main idea of the M-agent architecture starts with the following point: to design and realize multi-agent system we must first define what system component is supposed to be an agent, and what is not supposed to be ([12]). So the multi-agent system may be divided into two parts (Fig. 1):

- environment - that may be observed by the agent and represented by a corresponding model;
- agent's mind - an area in which the agent builds and processes an environment model.

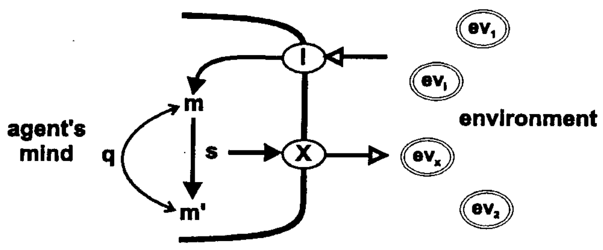


Figure 1: Principle of the M-agent architecture. I - imagination operation, X - execution operation, m, m' - models of the environment, s - strategy, q - evaluation function, ev_i - events taking place in the environment, ev_x - event produced as the result of the realization of the strategy s.

4.1.1 M-agent architecture

In the process of creating multi-agent system we consider that a given agent a remains and acts in an environment called V.
For any created agent a the following are defined:

- strategies S (s - strategy $s \in S$), goals Q (q - goal $q \in Q$), and models of the environment M (m - model of the environment, $m \in M$).
- operation of the observation I and operation of the strategy execution X.

The main idea of the agent functioning is the following:

- The agent observes the environment around and builds a model m of the environment in its mind. For this purpose it uses the observation (or imagination) operation I.
- The agent forecasts its possibilities using the strategy s. Applying the strategy s to the model m it obtains the model m' of the modified environment. Then, the agent compares in its mind the models m and m' using the function q that determines the goal of the agent functioning. It serves to select the best (from the point of view of the goal g) strategy s^*.
- If the evaluation of m and m' is satisfactory the agent realizes the selected strategy s^* that is represented by operation X - execution: $V' = X(s, V)$

So, in the rough, the first approach to the algorithm of a given agent may be expressed in three stages:

1. Observation - creation of the model m of the environment V:

$$m = I(M, V)$$

2. Reasoning - forecasting of the possible strategies application and evaluation of the result of the strategies application:
   - modeling of the strategy application $m' = s(m)$
   - evaluation of the strategy application: $q(m, s(m)) = q(m, m') \in \Re$
3. Decision - selection of the optimal strategy s^*:

$$s^* = F(M, Q, S, V)$$

   what may be realized by the following method: find s^* such that:

$$q(m, s^*(m)) = \max_{s \in S} q(m, s(m))$$

4. Action - realization of the optimal strategy s^* in the environment V:

$$V' = X(s, V)$$

where:
M, Q, S - sets of models, goals, strategies representing knowledge of the agent a,

V - environment (state),
I - observation function,
F - decision function enabling selection of the optimal strategy s^*,
X - function representing an operation of the realization (execution) of the agent's strategies.

The whole decision process is carried out by the processing of the models of the environment in the area called agent's mind.
The starting point to the agent oriented approach may be derived from the Object Oriented Analysis and Design and similarly expressed in the sentence:
"you are an agent, you must get along by yourself in the environment".
This is a useful starting point to define agents of the multi-agent system with the M-agent architecture concept.

4.1.2 Agent Model with Multiple Profiles

The agent model may be extended, so that we can consider a multi-profile M-agent architecture. Each profile is a M-agent model. T he agent observes the environment in each profile, and final decision is undertaken upon the results from each profile (Fig. 2).

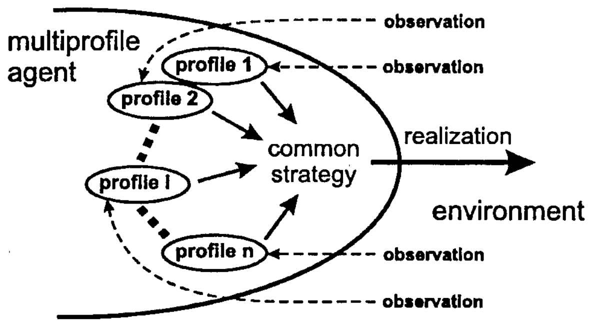


Figure 2: Concept of the multi-profile M-agent architecture

In a more complex case we can consider an agent with several profiles:

$$a = (a_1, a_2, ...a_i...a_n), \quad a_i = (M_i, Q_i, S_i, I, X, L)$$

where a_i - profile i of an agent a.
The operations I, X, L are common for all profiles, the configurations M_i, Q_i, S_i are different and characteristic of each profile. Each profile (egz. i) has defined its own strategy configuration S_i but the agent must realize only one strategy s belonging to the set S. This common configuration of strategies S is defined:

$$S \subset S_1 \times S_2 \times ... \times S_n, \quad s = (s_1, s_2, ...s_i, ...s_n)$$

$$\forall 1 \leq i \leq n \ \ s_i(m_i) = m_i'$$

The relation between the set of common goal Q and the goals in profiles $Q_i \forall 1 \leq i \leq n$ is defined as follows:

$$Q \subset Q_1 \times Q_2 \times ... \times Q_n, \quad q = (q_1, q_2, ...q_i, ...q_n)$$

$$\forall 1 \leq i \leq n \ \ q_i(m_i, m_i') \in \Re$$

where $\Re$ - real numbers.
The observation function I of the agent is defined as

$$(m_1, m_2, ...m_i, ...m_n) = I(M_1, M_2, ..., M_i, ..., M_n, V),$$

$$\forall 1 \leq i \leq n \ \ m_i \in M_i$$

The agent in every profile calculates the evaluation of a possible strategy $s = (s_1, s_2, ...s_i, ...s_n)$ by the value $DI_i(m_i, s) = q_i(m_i, s_i(m_i))$.
Final decision undertaken by the agent to realize the optimal strategy $s^* = (s_1^*, s_2^*, ...s_n^*)$ is carried out using the decision function U: $U(m, s^*) = \max_{s \in S} U(DI_1(m_1, s), DI_2(m_2, s), ...DI_n(m_n, s), AS)$ where AS - agent-state describes a general vital state of the agent and may be defined: $AS = \{m_1, m_2, ...m_i, ...m_n\}$ (m_i - just created model in the profile a_i of the environment observed).
The selected strategy s is realized by the agent in the environment V:

$$V' = X(s, V) = X((s_1, s_2, ...s_i, ...s_n), V)$$

4.1.3 Application of the M-agent architecture to the operational agent definition

Agent considers in its mind the result (m') of the application of the strategy s to the model m of the observed environment V ($m' = s(m)$). The modification of the model of the observed environment may be done by two ways:

- by changes of the environment (parameters of the environment)
- changing the point of view by agent displacement in the environment

Displacement of the agent makes that the observation of the environment V from the new agent position gives different model m'.
The mobile agent may be defined with the use of the M-agent architecture as a multiprofile agent in which at least one profile is the displacement profile.

An agent may execute operations on the environment by changing its parameters. The parameters of the environment may be considered as a set of resources that are processed by operations executed by the agent. So we can consider a kind of agent - processing agent, that when performing an operation, consumes one (kind) of given resource and produce another one. The model m of the environment contain information about consumed and produced resources before and after execution of the operation represented by the strategy s (Fig.4).

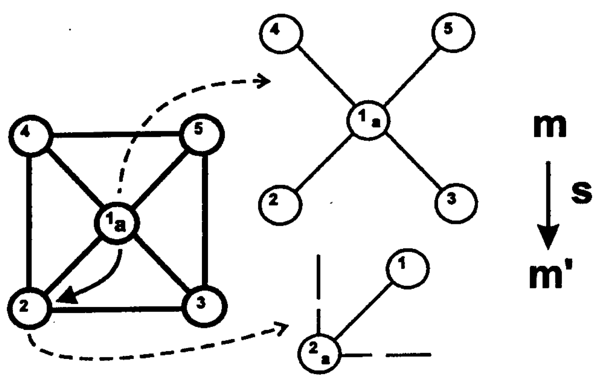


Figure 3: Principle of the agent displacement analyzed with the M-agent architecture: a - agent intending to displace from the node (1) to the node (2), m - model of the observed environment before displacement, m' - forecasted model of the environment to be observed after displacement.

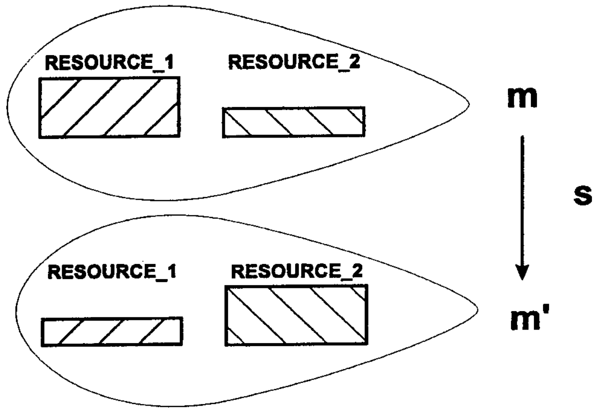


Figure 4: Principle of the operational profile of an agent

So the presented agent structure be considered as a profile of an agent and called *operational profile*. The operational agent, in the sense of M-agent architecture is the multiprofile agent with at least one operational profile.

Agents acting in the environment V have possibility to change this environment and adapt it to its own needs. This operation may be carried out by a group of agents that have the same (or similar) point of view. The decision to create new (sub) environment V_1 is undertaken by the agents due to the analysis of the common acting and benefit from that (for instance: cognition, consciousness and emergence theory [9])

The group of cooperating agents changes the part of the environment (by forming an organization) obtaining in this way a new environment V_1 (Fig. 5) that

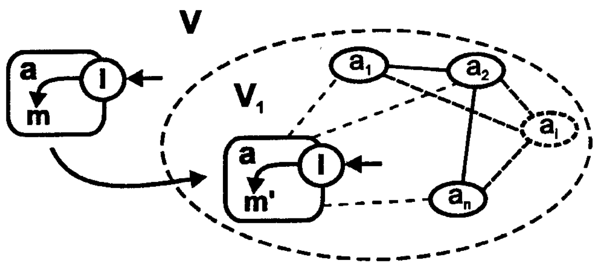


Figure 5: Principle of the grouping profile of an agent

may be considered as a sub-environment of V and called "organization". Then, due to the specialization and cooperation, they preserve and maintain the sub-environment V_1. Agent a entering an organization (team) is supposed to decide whether joining the organization V_1 is better than remaining in V (Fig. 5). The agent that has a profile to analyze the organization entering is an agent with grouping capability.

4.2 Operational agent architecture

Although the M-agent architecture can be easily extended with learning capabilities, it does not support directly implementation of cooperation mechanisms. In order to have agents being able to cooperate with others according to predefined cooperation mechanisms, we propose an extension of M-agent architecture to *the architecture of operational agent*.

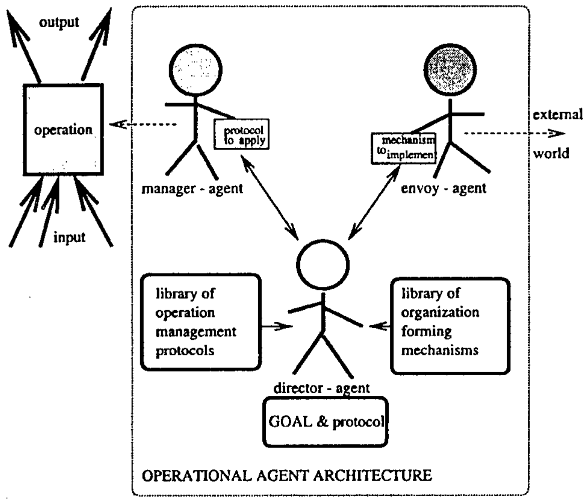
Operational agent is supposed to be bounded to some operation, see Fig. 6, and Fig. 7. The operation is either its own capabilities to perform some tasks, to produce some output from some input or expresses its expertise of management of such operation. Let us note that the operation itself may have internal structure consisting of sub-operational agents composed in a supply chain or a workflow.

The architecture of operational agent consists of three instances of M-agent: director-agent, manager-agent, and envoy-agent. The principal role of director agent is to determine protocols to follow for manager and envoy agents, and to coordinate their behaviors. The director agent has at its disposal:

- library of already constructed generic management protocols like MRP, or / and JIT, scheduling methods, etc.
- library of cooperation mechanisms, like bilateral contracts, CNP, auctions, and more sophisticated mechanisms for forming and reforming virtual organizations.

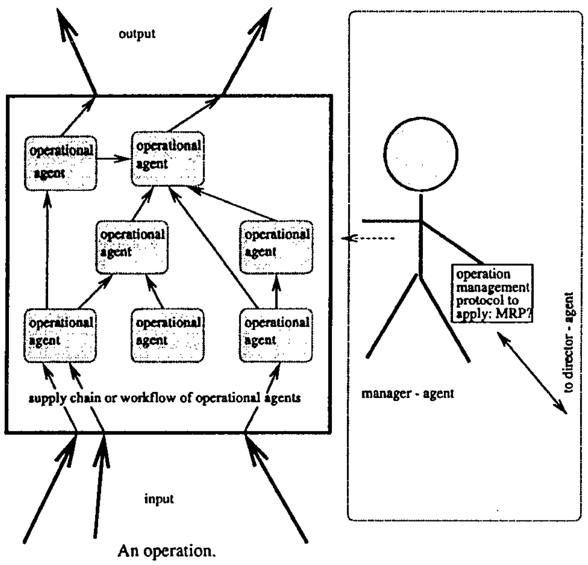
Manager-agent is responsible for internal affairs, i.e. for management of internal workflow or business process of the operation whereas the envoy-agent is responsible for external affairs, i.e. for input delivery and output distribution as well as for joining to already existing organizations or for making fusion with other agents, see [6, 7].

One example of functioning of operational agent is given in the next section.



The concept of operational agent:
manager, envoy, and director agents are instances of M-agent.

Figure 6:



An operation.

Figure 7:

5 Simple example of team formation

To illustrate the concept of operational agent architecture and the idea of organization formation, we present team formation process.

As the environment we consider a network of servers and mobile agents that can move from one server to another in order to look for resources scattered randomly on the servers. To enhance cooperation between the agents, the entry to some servers is forbidden for some agents. Thus, if a resource, wanted by an agent, is on a forbidden server, then the agent needs help from other agents. If several agents need the same resource, then they have to compete for it. Agents can execute the following types of actions: take (drop) a resource from (to) a server, receive (give) a resource from (to) another agent, move from one server to another, or to end his activity. The agents can communicate with each other only if they are at the same server. The set of mobile agents is denoted by M, whereas the set of the resources by R. Each mobile agent i is assigned an **original goal,** $G_i \subseteq R$, which consists in getting G_i with minimal effort and then to end his activity switching to an "off" state. The effort is related to agent's personal account, so that any action execution costs one unit, which is subtracted from the agent's account. Initially any agent is supposed to know only that his world is a computer network with agents living there however without specification. So that he does not know initially neither the map of the network nor the other agents and resource locations. An agent acquires knowledge from wandering over the network and from meeting other agents and gossiping with them.

If the network is large with a lot of agents looking and competing for the resources, then an agent has a little chance to realize his goal unless he tries to form a team of agents with consistent goals, i.e. the agents who do not compete with him for the same resources. Agents formed into a team extend their resources and capabilities to enter more servers in the network. However this is not the only gain from forming a team. Since the agents are realizing joint goal of the team, they have together greater computation power and more possibilities to get more and faster important information. Hence, by working in the team for realizing the joint goal, an agent increases his capabilities to achieve his individual goal.

The crucial point here is the notion of team. Team may be seen as an organization created by autonomous agents for fulfilling their joint goal. Since the environment is distributed, it forces all interactions inside a team, (i.e., information flow, decision making, goal realizing) to be local. Thus, in order to maintain integrity of the team, a mechanism is needed for eliminating inconsistent decision making as well as contradictory information and goals inside the team. To manage with all these problems we adopt the standard technique of token passing in a computer network. The token is the sign of decision power, so that a member of the team who has currently the token has got the exclusive authority to decide on the status of the team. The passing cycle of the token serves for the following purposes:

1. It determines the order in which the decision power is passed from one agent to another.
2. It determines the communication route.
3. It determines a plan of the cooperative work performed by a complete team.

The crucial idea of our formation mechanism relies on the fact that team is expanding locally, starting with a single agent. If the common goal of the expanding team becomes inconsistent according to the knowledge acquired from the dynamically changing environment, then the team starts shrinking by removing some of its members. As soon as the expanding team becomes complete, i.e., it has all the resources and capabilities needed to achieve the common goal, the members of the team perform the cooperative work. Another interesting feature of our approach is that the plan of the final cooperative work of the team is constructed and revised step by step during its formation.
However this example illustrates how to design efficient formation mechanisms, and how the basic organizational frames can be dynamically constructed and reconfigured in the formation process.

Team as a simple organization

The team structure and reconfiguration process is extremely simple and is based on the idea of token passing. The passing cycle serves for the following purposes:
1. For organizational frame, i.e. it determines the passing order of the decision power of the team.
2. For information flow frame, i.e. it assures the communication flow inside the team, and in this way it maintains the organizational integrity of the team.
3. For business frame, i.e. it assures that the cooperative work of the complete team is performed in the final mode.
The structure of team is flat and extremely simple. There are only two roles the agents can take, namely "team representative" for an agent who has currently the token, and "passive role" for agents not having the token at the moment. It is clear that passing the token according to the order of the passing cycle can not be effective. The reason is that the token should be rather passed to the agent who needs it for expanding or shrinking the team. The same concerns the cooperative work in the final mode. The resources should be passed directly to the agents who need them instead of wandering a long way along the passing cycle.

Operational agents in teams

Agents can be implemented according to the standard M-agent architecture. Operational agent is associated with the team token. So that there is one operational agent per team. In fact, the token carries the role of operational agent that is undertaken by the ordinary agent that has currently the token. The protocol for performing resources distribution in the final team mode refers to a management protocol. So that the team may be seen as an operation associated to that operational agent. The protocols to follow in expanding and shrinking modes refer to formation and reformation mechanisms. In this particular case operational agent may be seen as super mobile agent with the range being the sum of ranges of all members of the team. In fact, operational agent may be identified with the "mobile intelligent token" that uses an ordinary team member as its "incarnation" in order to use the expertise of that team member and to act in the range of that team member.
For more details concerning agent architectures, algorithms, implementation, and related work we refer to [2].

6 Conclusions

We have presented a novel architecture of middle agent called here operational agent architecture. Actually we apply this agent architecture in our ongoing projects on modeling agent virtual organizations and agent virtual workflow management. The organizations and workflow management systems are implemented on Agent Platform (http://www.ipipan.waw.pl/mas/) that supports network infrastructure construction.

References

[1] S. Ambroszkiewicz, and J. Komar. A model of BDI-agent in game-theoretic framework. In *Proc. ModelAge-97,* to appear in Springer LNAI. http://www.ipipan.waw.pl/mas/

[2] S. Ambroszkiewicz, O. Matyja, and W. Penczek. Team Formation by Self-Interested Mobile Agents. In Chengqi Zhang and Dickson Lukose (Eds.), *Proc. 4-th Australian DAI-Workshop,* Brisbane, Australia,

July 13, 1998. Published in Springer LNAI 1544. http://www.ipipan.waw.pl/mas/

[3] S. Ambroszkiewicz, O. Matyja, and W. Penczek. Cooperation Mechanisms in a Multi-Agent Distributed Environment. In *Proc. ECAI-98 Workshop w8,* Brighton, UK, August 24-28, 1998.

[4] S. Ambroszkiewicz, K. Cetnarowicz, T. Nowak, and B. Radko. Modelling Enterprise Formation and Integration as a Distributed Computing Performed by Mobile Agents. To appear in Proc. of HUNABC'98, First Hungarian National Conference on Agent Based Computing, May 29-31 (Fri-Sun) 1998, Budapest, Hungary.

[5] S. Ambroszkiewicz, K. Cetnarowicz, and B. Radko. Enterprise formation mechanism based on mobile agents. In Proc. of Intelligent Agents in Information and Process Management. A. Holsten, G. Joeris, C. Klauck, M. Klush, H.-Mueller, and J.P. Mueller (Eds.) Workshop at the 22nd German Annual Conference on Artificial Intelligence in Bremen (KI-98), TZI-Bericht Nr. 9, 1998. pp. 23-34.

[6] S. Ambroszkiewicz, K. Cetnarowicz, O. Matyja, and B. Radko. Modeling Virtual Enterprise: agent-based approach. In Proc. Multi Agent Systems Models Architecture and Applications. F. J. Garijo, and Ch. Lemaitre (Eds.), II Iberoamerican Workshop on D.A.I and M.A.S., October 1-2 1998 Toledo, Spain.

[7] S. Ambroszkiewicz. Agent Virtual Organizations within the Framework of Network Computing: a case study. In Proc. CEEMAS'99.

[8] Virtual Organizations research project sponsered by the Institute of Information Systems, Department of Information Management at the University of Berne. www.virtual-organization.net/

[9] N. A. Baas. Emergence, hierarchies and hyperstructures. *Artificial Life III, SFI studies in the science of complexity* Volume XVII, edited by C. G. Langton. Reading MA: Addison-Weseley, 1987.

[10] M. E. Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press, 1987.

[11] C. Castelfranchi and R. Falcone. From Task Delegation to Role Delegation. In Proc. AI*IA-97, LNAI 1321, 278-289.

[12] K. Cetnarowicz. M-agent architecture based method of development of multiagent systems. In *Proc. of the 8th Joint EPS-APS International Conference on Physics Computing*, ACC Cyfronet Krakow Poland, 1996.

[13] P.R. Cohen, H.J. Levesque, and Ira Smith. On Team Formation. In G. H. Hintikka and R. Tuomela (eds.), *Contemporary Action Theory,* Volume 2: Social Action, Synthese Library, Kluwer Academic Publishers, Dordrecht, Netherlands, 1997, pp. 87-114.

[14] K. Decker. Distributed Artificial Intelligence Testbeds. In G. M. P. O'Hare and N. R. Jennings (eds.): "Foundations of Distributed Artificial Intelligence , Sixth Generation Computer Technology Series, John Wiley & Sons, New York, 1996.

[15] Y. Demazeau and A.C. R. Costa "Populations and Organisations in Open Multi-Agent Systems", 1st Symposium on Parallel and Distributed AI, Hyderabad, India, July 1996.

[16] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In Proc. ICMAS-98.

[17] The Foundation for Intelligent Physical Agents (FIPA) http://drogo.cselt.stet.it/fipa/

[18] M.S. Fox et al. An Organizational Ontology for Enterprise Modeling. In M. Prietula et al.(Eds.). Simulating Organizations. pp. 131-152 AAAI Press and MIT Press, 1998.

[19] T. Ishida, M. Yokoo, and L. Gasser. An organizational Approach to Adaptive Production Systems. In *Proc. of 8th National Conf. on Artificial Intelligence,* Boston, USA, 1990.

[20] N. R. Jennings. Coordination Techniques for Distributed Artificial Intelligence. In G. M. P. O'Hare and N. R. Jennings (eds.): "Foundations of Distributed Artificial Intelligence, Sixth Generation Computer Technology Series, John Wiley & Sons, New York, 1996.

[21] B. Dunin-Keplicz and R. Verbrugge. Collective Commitments. In Proc. ICMAS'96, Kyoto, Japan, December 10-13, IEEE Computer Society Press 1996.

[22] D. Kinney, M. Ljungberg, A. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner (Eds.), *Artificial Social Systems, LNAI 830*, 1992.

[23] T. W. Malone, K. Crowston, J. Lee, B. Pentland, Ch. Dellarocas, G. Wyner, J. Quimby, Ch. Osborne, amd A. Bernstein. Tools for inventing organizations: Toward a handbook of organizational processes. Center for Coordination Science, MIT, http://ccs.mit/edu/

[24] T. W. Malone and K. Crowstone. Towards an interdisciplinary theory of coordination. *Computing Surveys,* 26(1), 87-119, 1994

[25] Mueller, J.P. The Design of Intelligent Agents. Springer LNAI 1177, 1996.

[26] S. Ossowski and A. Garcia-Serrano. Social Structure in Artificial Agent Societies. In Proc. ATAL-98.

[27] Peng Y., T. Finin, Y. Labrou, B. Chu. J. Long, W. T. Tollone, A. Boughannam. A Multi-Agent System for Enterprise Integration. In Proc. PAAM'98, The Practical Application of Intelligent Agents and Multi-Agent Systems. March 19998, London, UK.

[28] M. Prietula et al.(Eds.). Simulating Organizations. AAAI Press and MIT Press, 1998.

[29] A. S. Rao and M. P. Georgeff. Modelling rational agents within a BDI–architecture. In R. Fikes and E. Sandewall, editors, *Proc. of the 2rd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pp. 473–484, Cambridge, Mass., April 1991, Morgan Kaufmann.

[30] A. M. Ribeiro, and Y. Demazeau. A Dynamic Interaction Model for Multi-Agent Systems. In Proc. Multi Agent Systems Models Architecture and Applications. F. J. Garijo, and Ch. Lemaitre (Eds.), II Iberoamerican Workshop on D.A.I and M.A.S., October 1-2 1998 Toledo, Spain.

[31] T. Sandholm. Agents in Electronic Commerce: Component Technologies for Automated Negotiations and Coalition Formation. In *Proc. ICMAS'98*, pp. 10-11, Paris 1998.

[32] Smith, R. The Contract Net Protocol – High-Level Communication and Control in a Distributed Problem Solver. IEEE Trans. on Computers 29 (12), pp. 1104-1113.

[33] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research,* 7, 1997, pp. 83-124.

MULTI-AGENT SYSTEMS METHODOLOGY

Yves Demazeau

Leibniz IMAG, 46 avenue Felix Viallet, 38031 Grenoble cedex, France
Yves.Demazeau@imag.fr

(The paper is not available yet)

COLLECTIVE MOTIVATIONAL ATTITUDES IN COOPERATIVE PROBLEM SOLVING

Barbara Dunin-Kęplicz
Institute of Informatics
Warsaw University
Banacha 2, 02-097 Warsaw
Poland
E-mail: keplicz@mimuw.edu.pl

Rineke Verbrugge
Cognitive Science and Engineering
University of Groningen
Grote Kruisstraat 2/1, 9712 TS Groningen
The Netherlands
E-mail: rineke@tcw2.ppsw.rug.nl

Abstract

In this paper we investigate the role of collective commitments in groups of agents involved in Cooperative Problem Solving (CPS). Collective intentions and collective commitmens are formalized in the logical framework designed by Rao and Georgeff in [22]. The novelty of our approach is to base a collective commitment on a social plan, and defining it in terms of collective intentions and pairwise social commitments between team members together with mutual beliefs about them.

During social plan execution collective commitment within a group inevitably leads to the execution of agent-specific actions. However, due to the dynamic and possibly unpredictable environment, team members may fail their actions or be presented with new opportunities. Thus, it is necessary for them to monitor action performance and replan based on the present situation. This leads to the *reconfiguration problem*. In order to solve this problem, the main phases of CPS, namely construction, maintenance, and realization of collective commitments, are assigned to a four-level abstract architecture. In terms of these generic levels and their interplay *reconfiguration algorithm* is specified.

1 Introduction

Multiagent systems (MAS) are computational systems in which a collection of loosely-coupled autonomous agents interact in order to solve a given problem. As this problem is usually beyond the agents' individual capabilities, agents can communicate, cooperate, coordinate, and negotiate, both in advancement of their own goals as well as for the good of the system as a whole.

Some MAS are referred to as *intentional systems* realizing the practical reasoning paradigm ([1]). The best known and most influential are so-called *belief-desire-intention systems.* BDI-agents are characterised by a "mental state" described in terms of *beliefs*, corresponding to the information the agent has about the environment; *desires*, representing options available to the agent; and *intentions* representing the chosen options. While beliefs are viewed as agent's *informational* attitudes, desires as *goals*, intentions and commitments refer to its *motivational* attitudes.

In a formal specification of BDI-systems different kinds of modal logics are exploited. Dynamic, temporal and epistemic logics are extensively used to describe the single agent case. Inevitably, social and collective aspects of CPS should be investigated. Thus, from the perspective of teamwork, motivational attitudes are considered on three levels: individual, social (i.e. bilateral), and collective. In this paper we discuss and formalize social and collective aspects of intentions and commitments and their role in CPS. We agree with [21] that:

> Joint intention by a team does not consist merely of simultaneous and coordinated individual actions; to act together, a team must be aware of and care about the status of the group effort as a whole.

We aim at giving an internal (or prescriptive) theory of individual, social and collective motivational attitudes which allows agents to reason about cooperative problem solving (CPS). This is in contrast to external (or descriptive) theories which describe agents' attitudes from the perspective of the outside observer. In our approach, all individual motivational attitudes are viewed as primitive notions, but in contrast to [22] we investigate logical aspects of relations between individual intentions and social commitments on the one hand, and collective intentions and commitments on the other hand.

Starting from individual intentions, we first define the notion of a *collective intention* for a group. Our definitions are stronger than the ones introduced in [23], in particular a collective intention includes that all members *intend* for all others to share that intention. Together with individual and collective knowledge and belief, a collective intention constitutes a basis for preparing a plan (or a set of plans).

Planning may be done in many different ways; here we abstract from any particular methods, assuming only that a certain plan results, at the first place, in an adequate task division ensuring the realization of a collective goal. Next, an appropriate allocation of tasks (goals) to group members should be established. Both steps together reflect an operational way to achieve the goal in question.

Next, we characterize the strongest motivational attitude, namely the *collective commitment* of a group. We assume that bilateral aspects of a plan — mutual obligations between agents — are reflected in *social commitments.* Thus collective commitments are defined on the basis of collective intentions and social commitments. A collective commitment is meant to reflect the way a plan is to be executed, but one should take into account that agents are autonomous entities taking individual decisions in the light of the current situation and the commitment strategies they follow. We provide a classification of agents' commitment strategies based on an analogical characterization of intentional strategies given by Rao and Georgeff in [22]. Let us stress, however, that because the change of an agent's social commitment directly influences the way the plan is executed, the conditions imposed on agents dropping social commitments are more restrictive than in the case of changing individual intentions.

The theory of collective motivational attitudes, with collective commitment as a central concept, focuses on static aspects of CPS. The proper treatment of collective commitments in a dynamically changing environment entails the maintenance of all individual, social and collective motivational attitudes involved throughout the whole process. We isolate four basic stages involved in teamwork in a distributed environment. These are construction, maintenance, and realization of collective commitments. We base our analysis on the four-stage model of [31], containing the consecutive stages of *potential recognition*, *team formation*, *plan formation* and *team action.* These stages will be viewed as levels of abstraction and constitute together an abstract architecture. Taking into account the unpredictable environment, all four stages have a dynamic character and require methods reflecting this. When defining the levels we abstract from particular methods and algorithms meant to realize level-associated goals, but instead formulate their final results and associate them with appropriate individual, social, and collective motivational attitudes.

Even though potential recognition, team formation and plan formation have been extensively discussed in the MAS and AI literature, the important phase of collective team action has received relatively little attention. The requirements of a constantly changing environment lead to the *reconfiguration problem*: when maintaining a collective intention during plan execution, it is crucial that agents replan properly and efficiently when some members do not fulfill their delegated subtasks or are presented with new opportunities. This problem may be treated as the core of plan execution. It has been the main subject of [12, 11, 13]: in terms of the generic levels and their interplay an efficient and flexible *reconfiguration algorithm* has been specified. The algorithm description is meant to be generic: a pattern of behavior is described in terms of abstract level-associated procedures, viewed as a sort of black box with specified input and output parameters. The next step should be a transformation of *black* boxes into *glass* ones.

The paper is structured in the following manner. In section 2, notions of a group and collective belief are introduced. In sections 3 and 4 some motivational attitudes like individual goals and intentions and social commitments are discussed as well as different types of agents' commitment strategies. On this basis collective motivational attitudes are defined in section 5. Section 6 presents the four-level description of the abstract system. In terms of these generic abstract levels a general-purpose reconfiguration algorithm is defined in section 7. Finally, the last sections focus on discussion and options for further research.

2 Beliefs

To represent beliefs, we adopt a standard $KD45_n$-system for n agents as explained in [15], where we take $\mathrm{BEL}(a, \varphi)$ to have as intended meaning "agent a believes proposition φ". $KD45_n$ consists of the following axioms and rules for $i = 1, \ldots, n$:

A1 All instantiations of tautologies of the propositional calculus

A2 $\mathrm{BEL}(i,\varphi) \wedge \mathrm{BEL}(i,\varphi \rightarrow \psi) \rightarrow \mathrm{BEL}(i,\psi)$ (Belief Distribution Axiom)

A4 $\mathrm{BEL}(i,\varphi) \rightarrow \mathrm{BEL}(i,\mathrm{BEL}(i,\varphi))$ (Positive Introspection Axiom)

A5 $\neg\mathrm{BEL}(i,\varphi) \rightarrow \mathrm{BEL}(i,\neg\mathrm{BEL}(i,\varphi))$ (Negative Introspection Axiom)

A6 $\neg\mathrm{BEL}(i,\perp)$ (Consistency Axiom)

R1 From φ and $\varphi \rightarrow \psi$ infer ψ (Modus Ponens)

R2 From φ infer $\mathrm{BEL}(i,\varphi)$ (Belief Generalization)

In the semantics, there are accessibility relations B_i that lead from worlds w to worlds that are consistent with agent i's beliefs in w. Thus, BEL is defined semantically as follows:
$w \models \mathrm{BEL}(i,\varphi)$ iff
$t \models \varphi$ for all t such that wB_it.

Note that the B_i need not be reflexive, corresponding to the fact that an agent's beliefs need not be true. On the other hand, the accessibility relations B_i are transitive, euclidean and serial. These conditions correspond to the axioms of positive and negative introspection and to the fact the agent has no inconsistent beliefs, respectively. It has been proved that $KD45_n$ is sound and complete with respect to these semantics. (The property of negative introspection is controversial; we are agnostic about this and dropping [A5] will not have important consequences for the logical framework presented in this paper.)

One can define modal operators for group beliefs. The formula $\text{E-BEL}_G(\varphi)$ is meant to stand for "every agent in group G believes φ". It is defined semantically as $w \models \text{E-BEL}_G(\varphi)$ iff for all $i \in G$, $w \models \mathrm{BEL}(i,\varphi)$), which corresponds to the following axiom:

C1 $\text{E-BEL}_G(\varphi) \leftrightarrow \bigwedge_{i\in G} \mathrm{BEL}(i,\varphi)$

A stronger operator is the one for collective belief, which is similar to the more usual common knowledge. $\text{C-BEL}_G(\varphi)$ is meant to be true if everyone in G believes φ, everyone in G believes that everyone in G believes φ, etc. Let $\text{E-BEL}^1_G(\varphi)$ be an abbreviation for $\text{E-BEL}_G(\varphi)$, and let $\text{E-BEL}^{k+1}_G(\varphi)$ for $k \geq 1$ be an abbreviation of $\text{E-BEL}_G(\text{E-BEL}^k_G(\varphi))$. Thus we have $w \models \text{C-BEL}_G(\varphi)$ iff $w \models \text{E-BEL}^k_G(\varphi)$ for all $k \geq 1$. Note that even collective beliefs need not be true, so $w \models \text{C-BEL}_G(\varphi)$ need not imply $w \models \varphi$. Define t to be *G-reachable* from s if there is a path in the Kripke model from s to t along accessibility arrows B_i that are associated with members i of G. Then the following property holds (see [15]):
$s \models \text{C-BEL}_G(\varphi)$ iff $t \models \varphi$ for all t
that are G-reachable from s.

Using this property, it can be shown that the following axiom and rule can be soundly added to the union of $KD45_n$ and [C1]:

C2 $\text{C-BEL}_G(\varphi) \rightarrow \text{E-BEL}_G(\varphi \wedge \text{C-BEL}_G(\varphi))$

RC1 From $\varphi \rightarrow \text{E-BEL}_G(\psi \wedge \varphi)$ infer $\varphi \rightarrow \text{C-BEL}_G(\psi)$ (Induction Rule)

The resulting system is called $KD45^C_n$, and it is sound and complete with respect to Kripke models where all n accessibility relations are transitive, serial and euclidean [15].

Some of the ways in which individual beliefs can be generated are updating, revision, and contraction. The establishment of collective beliefs among a group is more problematic. In [18] it is shown that bilateral sending of messages does not suffice to determine collective belief if communication channels may be faulty, or even if there is uncertainty whether message delivery may have been delayed. We assume that in our groups, a more general type of communication, e.g. by a kind of global announcement, can be achieved. A good reference to the problems concerning collective belief and to their possible solutions is [15, Chapter 11]. In any case, it is generally agreed that collective belief is a good *abstraction tool* to study teamwork.

3 Individual motivational attitudes

Our approach to describe motivational attitudes and related aspects is minimal in the sense that we aim to deal with concise necessary and sufficient conditions. Additional aspects appearing on the stage in specific cases may be addressed by refining the system and adding new axioms. After an introduction of the notation, this section focuses on individual goals and intentions, and gives a short overview of our choice of axioms and corresponding semantic conditions from [22].

3.1 Notation

In our framework, most axioms relating motivational attitudes of agents appear in two forms: one with respect to *propositions*, the other with respect to *actions*. These actions are interpreted in a generic way — we abstract from any particular form of actions: they may be complex or primitive, viewed traditionally with certain effects or with default effects [6, 7, 8], etc.

A proposition, on the other hand, reflects the particular state of affairs that an agent aims for. In other words, propositions represent the agent's higher level goals. Again, we abstract from particular methods of achieving them; e.g. they may be realized by particular plans.

Table 1 gives the formulas appearing in this paper, together with their intended meanings.

COMM(a, b, φ)	agent a commits to agent b to make φ true
COMM(a, b, β)	agent a commits to agent b to do β
GOAL(a, φ)	agent a has as a goal that φ be true
GOAL(a, β)	agent a has as a goal to do β
$stit(a, \varphi)$	agent a sees to it that φ holds
$done(a, \beta)$	agent a has done β at the previous moment
INT(a, φ)	agent a has the intention to make φ true
INT(a, β)	agent a has the intention to do β
E-INT$_G(\varphi)$	every agent in group G has the intention to make φ true
E-INT$_G(\beta)$	every agent in group G has the intention to do β
C-INT$_G(\varphi)$	group G has the collective intention to make φ true
C-INT$_G(\beta)$	group G has the collective intention to do β
C-COMM$_{G,P}(\varphi)$	group G has a collective commitment to make φ true by plan P
C-COMM$_{G,P}(\beta)$	group G has a collective commitment to do β by plan P

Table 1: Formulas and their intended meaning

The symbol φ denotes a proposition and β an action. Even though it may seem from the table as if the formulas have only an informal meaning (perhaps derived from so-called folk psychology), this is actually not the case. In fact, the individual motivational attitudes are primitive but are governed by axiom systems and corresponding semantics, while the social and collective motivational attitudes are defined by axioms in terms of the individual ones.

3.2 Individual goals and intentions

In Rao and Georgeff's [22], individual beliefs, goals, and intentions are formalized as primitive notions and given a formal semantics. We take their semantics as a basis for our formalization of collective motivational attitudes, and refer the reader to [22, 24] for details, especially completeness and decidability proofs. However, we give a short description here, providing the background needed to follow this paper.

As a reminder, the temporal structure is a discrete tree branching towards the future, as in Computation Tree Logic CTL, which is used for studying concurrent programs (see [14] for a semantic and axiomatic treatment). The different branches in such a time tree denote the optional courses of events that can be chosen by an agent. Primitive events are those events that an agent can perform and that determine the next point in the time tree. The branch between a point and the next point is labeled with the primitive event leading to that point. For example, if there are two branches emanating from a single time point, one labeled "go to dentist", and the other "go shopping", then the agent has a choice of executing either of these events and moving to the next point along the associated branch.

The temporal operators include *inevitable*(φ) (in all paths through the point of reference φ holds), *optional*$(\varphi) \equiv \neg$*inevitable*$(\neg\varphi)$, $\Diamond\varphi$ (somewhere later on the same path, φ holds) and φ **U** ψ (φ *until* ψ, i.e. either φ holds forever on this path, or, as soon as it stops holding, ψ will hold). Formulas are divided into state formulas (which are true in a particular state) and path formulas (which are true along a certain path). Here follows our definition, which adapts Rao and Georgeff's single-agent definition to the n-agent case with a set of agents A:

S1 each atomic proposition is a state formula;

S2 if φ and ψ are state formulas, then so are $\neg\varphi$ and $\varphi \wedge \psi$;

S3 if φ is a path formula then *inevitable*(φ) and *optional*(φ) are state formulas;

S4 if φ is a state formula, then so are $\mathrm{BEL}(a, \varphi)$, $\mathrm{GOAL}(a, \varphi)$ and $\mathrm{INT}(a, \varphi)$ for all $a \in A$;

P0 if φ and ψ are state formulas, then $\Diamond\varphi$ and φ **U** ψ are path formulas.

As to Kripke semantics, we consider each possible world to be a temporal tree structure as described above with a single past and branching time future. Evaluation of formulas is with respect to a world w and a state s, using ternary accessibility relations B_i, D_i and I_i on corresponding to each agent's beliefs, goals (or desires), and intentions, all of which lead from a pair of a world and a state in it to a world. Evaluation of formulas at world-state pairs is defined in the obvious manner inspired by CTL and epistemic logic. Here we give only our n-agent adaptation of the definitions for goals and intentions, where the expression $M, w_s \models \varphi$ is read as "formula φ is satisfied by world w and state s in structure M". For $i = 1, \ldots, n$ we have:

$M, w_s \models \mathrm{GOAL}(i, \varphi)$ iff
$\forall v$ with $(w, s, v) \in D_i$, $M, v_s \models \varphi$

$M, w_s \models \mathrm{INT}(i, \varphi)$ iff
$\forall v$ with $(w, s, v) \in I_i$, $M, v_s \models \varphi$

The full definition of formula evaluation can be found in [24], and some examples are given in [22]. We will need this notion of possible worlds throughout the rest of this paper.

Rao and Georgeff [24] give an axiomatization of a basic BDI-logic for the single-agent case, which includes all CTL-axioms for the temporal component. For the epistemic operator BEL the modal system $KD45$ is used, which we adapt to $KD45_n$ for n agents (see the previous section). For the motivational operators GOAL and INT the axioms include the system K, which we adapt for n agents to K_n. For $i = 1, \ldots, n$ the following axioms and rules are included:

A1 All instantiations of tautologies of the propositional calculus

R1 From φ and $\varphi \rightarrow \psi$ infer ψ (Modus Ponens)

A2$_D$ $(\mathrm{GOAL}(i,\varphi) \wedge \mathrm{GOAL}(i,\varphi \rightarrow \psi) \rightarrow \mathrm{GOAL}(i,\psi)$
(Goal Distribution Axiom)

A2$_I$ $(\mathrm{INT}(i,\varphi) \wedge \mathrm{INT}(i,\varphi \rightarrow \psi) \rightarrow \mathrm{INT}(i,\psi)$
(Intention Distribution Axiom)

R2$_D$ From φ infer $\mathrm{GOAL}(i,\varphi)$
(Goal Generalization)

R2$_I$ From φ infer $\mathrm{INT}(i,\varphi)$
(Intention Generalization)

In a multi-agent system, an agent starts from goals. As an agent may have many different objectives, its goals need not be consistent with each other. Then, the agent chooses a limited number of its goals to be intentions, which are chosen in such a way that consistency is preserved. Thus for intentions Rao and Georgeff assume, as we do, that they should be consistent. This can be formulated as follows:

A6$_I$ $\neg\mathrm{INT}(i,\perp)$ for $i = 1,\ldots,n$
(Intention Consistency Axiom)

Rao and Georgeff also add an analogous axiom for the consistency of goals. However, it was argued above that an agent's goals are not necessarily consistent with each other. Thus, we adopt the basic system K_n for goals. Nevertheless, in the presented approach other choices may be adopted without consequences for the rest of the definitions in this paper.

In order to formulate the axiom that captures the fact that the set of an agent's intentions is a subset of its goals, we will first give a semantic intuition. Because intentions are chosen goals, one would expect the trees that are intention-accessible for an agent to be narrower versions of the trees that are goal-accessible; after all, some optional courses of events have been left out by the agent when going from goals to intentions. This semantic condition is formulated more precisely below. At first sight one would formulate the fact that intentions are chosen goals as $\mathrm{INT}(i,\varphi) \rightarrow \mathrm{GOAL}(i,\varphi)$ for all formulas. However, for some strange formulas this formula does not hold on the intended models, for example $\mathrm{INT}(i, \mathit{inevitable}(\varphi)) \rightarrow \mathrm{GOAL}(i, \mathit{inevitable}(\varphi))$ need not hold when the intention-accessible worlds are narrower versions of the goal-accessible ones. For, $\mathit{inevitable}\varphi$ says that φ should hold on all branches through the point of evaluation; thus $\mathit{inevitable}\varphi$ may hold on the narrower tree of intention-accessible worlds but not on its supertree of goal-accessible worlds.

Thus, a syntactic definition is needed to formulate a restricted version of the intention-goal compatability axiom. O-formulas are defined to be those formulas that contain no positive occurrences of *inevitable* (this is a somewhat stronger definition than Rao and Georgeff's, but easier to understand and use). The intuition behind O-formulas is that their truth is preserved when going from a narrower tree to its supertrees. Note that $\mathit{inevitable}(\varphi)$ is not an O-formula, and indeed, as was shown above, its truth is not always preserved when going from a narrower tree to its supertree.

For φ an O-formula, we adopt Rao and Georgeff's axiom of goal-intention compatibility, which belongs to the multi-modal extension of their basic BDI axiom system. Thus, adapting to the n-agent case, intentions are stronger than goals: they are chosen goals.

ID-SA1 $\mathrm{INT}(i,\varphi) \rightarrow \mathrm{GOAL}(i,\varphi)$
for $i = 1,\ldots,n$.

This axiom corresponds to the following semantic condition: for all w' which are goal-accessible for agent i from (w,s), there is a w'' which is intention-accessible for agent i from (w,s), and which is a *subtree* of w'. Intuitively, a world w'' is a subtree of w' if w'' contains fewer paths, but is otherwise identical to w' (see RG95).

Rao and Georgeff prove soundness and completeness of their basic BDI-logic and some extensions with respect to suitable classes of models by a tableau method, and also give decidability results using a small model theorem. It is easy to see that their methods can be extended to our choice of axioms for the n-agent case.

4 Social commitments

As [2] showed, it is important to distinguish between individual intentions, bilateral commitments, and collective motivational attitudes. A social commitment between two agents is not as strong as a collective commitment among them (see subsection 5.2), but stronger than an individual intention of one agent. If an agent commits to a second agent to do something, then the first agent should have the *intention* to do that. Moreover, the first agents commits to the second one only if the second one is *interested* in the first one fulfilling its intention. These two conditions are inspired by [2], but we find that for a social commitment to arise, a third condition is necessary, namely that the agents are aware about the situation, i.e. about their individual attitudes. Such awareness, expressed in terms of collective belief, is generally achieved by communication:

$$\begin{aligned}&\mathrm{COMM}(a,b,\varphi) \leftrightarrow \mathrm{INT}(a,\varphi)\wedge\\ &\quad \mathrm{GOAL}(b, \mathit{stit}(a,\varphi)) \wedge\\ &\quad \mathrm{C\text{-}BEL}_G(\mathrm{INT}(a,\varphi) \wedge \mathrm{GOAL}(b, \mathit{stit}(a,\varphi)))\end{aligned}$$

where $\mathit{stit}(a,\varphi)$ means that agent a sees to it (takes care) that φ becomes true (see [26]).

4.1 Commitment strategies

The key point is whether and in which circumstances an agent can drop a social commitment. If such a situation arises, the next question is how to deal with it responsibly.

We define three kinds of agents according to the strength with which they maintain their social commitments. The definitions are inspired by those in [22] for intention strategies. The need for agents' responsible behaviour led us to include additionally the social aspects of communication and coordination. We assume that the commitment strategies are an immanent property of the individual agent and that they do not depend on the goal to which the agent is committed, nor on the other agent to whom it is committed.

We also assume that each agent knows which commitment strategies are adopted by itself and by other agents in the group. This meta-knowledge ensures proper replanning and coordination as was discussed in [10].

The strongest commitment strategy is followed by the *blindly committed* agent, who maintains its commitments until it actually believes that they have been achieved. Formally,

$$\begin{aligned}&\mathrm{COMM}(a,b,\varphi) \rightarrow\\ &\quad \mathit{inevitable}[\mathrm{COMM}(a,b,\varphi)\ \mathrm{U}\\ &\quad \mathrm{BEL}(a,\varphi)]\end{aligned}$$

Single-minded agents may drop social commitments when they do not believe anymore that the commitment is realizable. However, as soon as the agent abandons a commitment, some communication and coordination with the other agent is needed. For open-minded agents, the situation is similar as for single-minded ones, except that they can also drop social commitments if they do not aim for the respective goal anymore. As in the case of single-minded agents, communication and coordination will be involved as expressed by the axiom:

$$\begin{aligned}&\mathrm{COMM}(a,b,\varphi) \rightarrow\\ &\quad \mathit{inevitable}[\mathrm{COMM}(a,b,\varphi)\ \mathrm{U}\\ &\quad \{\mathrm{BEL}(a,\varphi) \vee\\ &\quad (\neg\mathrm{BEL}(a, \mathit{optional}\ \Diamond\varphi) \wedge\\ &\quad \mathit{done}(\mathit{communicate}(a,b,\neg\\ &\quad \mathrm{BEL}(a, \mathit{optional}\ \Diamond\varphi))) \wedge\\ &\quad \mathit{done}(\mathit{coordinate}(a,b,\varphi)))\\ &\quad \vee(\neg\mathrm{GOAL}(a,\varphi) \wedge\\ &\quad \mathit{done}(\mathit{communicate}(a,b,\neg\mathrm{GOAL}(a,\varphi))) \wedge\\ &\quad \mathit{done}(\mathit{coordinate}(a,b,\varphi)))\}].\end{aligned}$$

5 Collective motivational attitudes

Groups are created on the basis of *collective intentions*, which are defined in subsection 5.1. However in this paper we abstract from the ways in which groups are formed, and refer the interested reader to [5, 19, 31]. The behaviour of such groups will be studied with respect to both collective intentions and collective commitments, which are defined in subsection 5.2. The creation of a collective commitment is based on the corresponding collective intention and hinges on the allocation of actions according to an adopted plan. However, some agents in the group may not have delegated actions while still being involved in the collective intention and the collective commitment. The group exists as long as the collective intention among them exists.

When creating a collective commitment, the group as a whole is known, including the meta-knowledge about the members' commitment strategies. For simplicity we assume that an agent's commitment strategy persists during plan realization, after which it is allowed to change its strategy.

The reader will note that collective intention and collective commitment are not introduced as primitive modalities, with some restrictions on the semantic accessibility relations (as in e.g. [3]). We do give necessary and sufficient conditions for such collective motivational attitudes to be present. In this way, we hope to make the behavior of a team easier to predict.

In the philosophical and MAS literature there is an ongoing discussion whether collective intentions may be reduced to individual ones plus collective beliefs about them (see [29, 2, 17]). Even though our definition seems to be reductive, it involves nested intentions and collective epistemic operators, and for this reason is deeper than a simple compound built out of individual intentions and collective beliefs about them by propositional connectives only.

Let us stress again that we have tried to find *minimal* conditions for collective intentions and collective commitments to be present, and not to weigh down the definitions with all aspects that play a part in the establishment of collective intentions and commitments. Such elements as conventions, abilities, opportunities, power relations and social structure (see [27, 28, 31] for a thorough discussion) certainly are important, and we leave open the possibility of defining and using them in specific cases where they play a crucial role, for example in the form of additional axioms.

5.1 Collective intentions

In order to establish a collective intention among the members of a group, a necessary condition is that all members of the group have the associated individual intention, and that it is a collective belief in the group that all members have this intention, i.e. all members of the group are aware of this. In fact, this condition is taken to give the full definition of collective intention in [23] (see [31] for a similar definition of collective goal). However, this condition is certainly not sufficient. Imagine that two agents want to achieve the same goal but are in a competition in which both want to be the only one to achieve the goal. Suppose also that it is a collective belief among the two agents that both want to achieve this goal. In this case the agents cannot be said to cooperate: they do not have a collective intention, even though the necessary condition stated above is fulfilled.

This example suggests adding an extra condition in the definition of a collective intention: all members in the group should *intend* the other members to have the associated individual intention ; and it should be a collective belief in the group that this is so.

In order to formalize the above two conditions for collective intentions among a group G, let us recall that φ stands for a proposition and α for an action. The formulas E-INT$_G(\varphi)$ and E-INT$_G(\alpha)$ (standing for "everyone intends") are syntactically defined in a similar way as the operator for "everyone believes", namely by the following axiom:

$$\text{E-INT}_G(\varphi) \leftrightarrow \bigwedge_{i \in G} \text{INT}(i, \varphi)$$

Now we are ready to adopt the following axiom defining collective intentions (the analogous axioms for actions α hold as well):

$$\begin{aligned}&\text{C-INT}_G(\varphi) \leftrightarrow \\ &\quad \text{E-INT}_G(\varphi) \wedge \text{C-BEL}_G(\text{E-INT}_G(\varphi)) \\ &\quad \wedge \text{E-INT}_G(\text{E-INT}_G(\varphi)) \\ &\quad \wedge \text{C-BEL}_G(\text{E-INT}_G(\text{E-INT}_G(\varphi)))\end{aligned}$$

5.2 Collective commitments

Inspired by [2], we treat collective commitment as the strongest motivational attitude to be considered in teamwork. In our opinion a collective intention is a necessary but not sufficient condition for a collective commitment to be present. A collective intention may be viewed as an inspiration for team activity, whereas the collective commitment reflects the concrete manner of achieving the intended goal by the group. This concrete manner is provided by planning. Thus, our approach to collective commitments is plan-based.

5.2.1 Social plans

We see planning, including means-ends analysis, as a two-step process. The first step is *task division* or decomposition, in which the question is adressed how to decompose a complex task φ into (possibly also complex) subtasks $\varphi_1, \dots, \varphi_n$. This step is followed by *task allocation*, in which actions are associated to the subtasks constructed in task division, and these are given to team members. This results in pairs $< \alpha_i, b >$ of a possibly complex action α_i that realizes task φ_i and an agent b. To make a social plan complete, the temporal structure among these pairs should be established. This is reflected in the recursive definition of a *social plan expression*, adopted from [23] and inspired by dynamic logic. The plans on which collective commitments are based are always represented as social plan expressions.

- If p is an atomic action and y is an agent, then $< p, y >$ is a well-formed social plan expression;
- If φ is a well-formed state formula and y is an agent, then $<!\varphi, y>$ (standing for y to *achieve* φ) and $<?\varphi, y>$ (standing for y to *test* the truth of condition φ) are well-formed social plan expressions;
- If α and β are well-formed social plan expressions, then $< \alpha;\beta >$ (sequential composition), $< \alpha \parallel \beta >$ (parallellism) and $< \alpha \mid \beta >$ (non-deterministic choice) are well-formed social plan expressions.

The last part of the definition introduces the temporal relations between the execution of actions. Paradigmatic aspects like negotiation, communication and coordination are all involved in planning. The result of the whole planning process is a plan P, represented as a social plan expression.

5.2.2 The definition of a collective commitment

Let us turn to a definition of collective commitment based on the plan P. A collective commitment among a team can only be established or maintained if the group has the associated collective intention.

In addition, for every one of the subgoals $\varphi_1, \dots, \varphi_n$ established during task division that together constitute the overall goal φ in question, there should be one agent in the group who is socially committed to at least one (mostly other) agent in the group to fulfil the subgoal. Moreover, there should be a collective belief in the whole group that the plan will be entirely realised, i.e. that all actions have been adopted by committed members of the group. The defining axiom below reflects all these characteristics.

$$\begin{aligned}
&\text{C-COMM}_{G,P}(\varphi) \leftrightarrow \text{C-INT}_G(\varphi) \wedge \\
&\quad \bigwedge_{\varphi_i \in P} \bigvee_{a,b \in G} \text{COMM}(a, b, \varphi_i) \wedge \\
&\quad \text{C-BEL}_G\Big(\bigwedge_{\varphi_i \in P} \bigvee_{a,b \in G} \text{COMM}(a, b, \varphi_i)\Big)
\end{aligned}$$

Unlike other definitions of joint or collective motivational attitudes (see e.g. [21, 31]), this definition of collective commitments is easy to understand and to use. Because the definition is part of an internal theory, it has pragmatic power: agents can take the whole process of building and revising collective commitments into their own hands.

Let us stress that this definition is parsimonious: we consider social commitments and collective beliefs about them between pairs of agents, instead of among the team as a whole. It is not assumed that all agents in G know the plan P. Let us stress that in this definition the team as a whole is aware that social commitments between team members have been established in the proper way, without explicit knowledge of each bilateral commitment. This kind of explicit knowledge can be incorporated when needed. Also, some other approaches to collective commitments (see e.g. [31]) consider triggers for commitment adoption formulated as preconditions. If needed, these may be incorporated into our framework as well by adding an extra axiom.

Our approach is especially strong when replanning is needed. In fact, in a constantly changing environment it often occurs that during action execution some agents do not fulfil their delegated actions. The problem of replanning properly and efficiently in such circumstances is called the *reconfiguration problem*, which will be treated in the next two sections. In contrast to [31], using our definition of collective commitment it is often sufficient to revise some of the pairwise social commitments, instead of involving the entire team in the replanning process. This is a consequence of basing our definition of collective commitment on an explicitly represented plan, and of building it from pairwise social commitments. In effect, if the new plan resulting from the analysis of the current situation within the team and the environment is as close as possible to the original one, the process of replanning is maximally efficient. Note that in contrast to other approaches ([31], [21]), the collective commitment is not iron-clad: it may vary in order to adapt to changing circumstances, so that the collective intention on which it is based can still be reached.

Even though the definition of collective commitment is intuitive, its complexity calls for a rigorous maintenance of all motivational attitudes involved in CPS, especially during reconfiguration. This is the subject we turn to now.

6 The four levels of CPS

We base our analysis of CPS on the four-stage model of [31], containing the consecutive stages of *potential recognition*, *team formation*, *plan formation* and *team action*. These stages will be viewed as levels of abstraction and constitute together an abstract architecture. Taking into account the unpredictable environment, all four stages have a dynamic character and require methods reflecting this. When defining the levels we abstract from particular methods and algorithms meant to realize level-associated goals, but instead formulate their final results and associate them with appropriate individual, social, and collective motivational attitudes.

Level 1: potential recognition

The starting point of the cooperative problem solving process is the overall goal that should be achieved. In this paper, as in [31], we restrict ourselves to the situation with one fixed overall goal and one agent that takes the initiative to achieve that goal. Analogous to [31], we consider CPS to begin when some agent in a multiagent environment recognizes the potential for cooperative action to reach the overall goal φ of the system.

Input and output of potential recognition

The input of the potential recognition stage is an agent a, a goal φ plus a finite set T of agents from whom a potential team may be formed.

The output at this stage is the "potential for cooperation" that the initiator a sees with respect to φ, denoted as $\text{POTCOOP}(a, \varphi)$, meaning that:

- φ is a goal of a ($\text{GOAL}(a, \varphi)$);
- either a cannot achieve φ by itself, or it believes that it does not have any (complex) action at its disposal which realizes φ and which it has as a goal to execute in isolation:

  $\neg\text{CAN}(a, \varphi)\vee$
  $\text{BEL}(a, \neg\exists\beta[\text{AGENTS}(\beta, \{a\})\wedge$
  $\text{MEANS-FOR}(\beta, \varphi) \wedge \text{GOAL}(a, \beta)])$

- there is a group G such that a believes that:
  - G can collectively achieve φ ($\text{C-CAN}_G(\varphi)$),
  - members of G have the right distribution of commitment strategies to do so ($\text{STRAT}(\varphi, G)$)
  - they are willing to participate in team formation ($\forall i \in G willing(i, \varphi)$)

Now, we are ready to present the formal definition of potential for cooperation with respect to the overall goal φ. All formulas that are used in the definition are presented informally in table 2.

KNOW(a, φ)	agent a knows that φ
AGENTS(β, G)	group G is precisely the set of agents required to perform the action β
MEANS-FOR(β, φ)	the (possibly complex) action β is a means to realize the goal φ
CAN(a, φ)	agent a can individually achieve goal φ
C-CAN$_G(\varphi)$	group G can collectively achieve goal φ
willing$(i, \varphi))$	agent i is willing to participate in team formation with respect to goal φ
STRAT(φ, G)	group G have a suitable distribution of commitment strategies to achieve φ.

Table 2: Formulas needed for the definition of potential recognition

$$\begin{aligned}
&\text{POTCOOP}(a,\varphi) \leftrightarrow \text{GOAL}(a,\varphi) \wedge \\
&\quad \{\neg \text{CAN}(a,\varphi) \vee \\
&\quad \text{BEL}(a, \neg\exists\beta[\text{AGENTS}(\beta,\{a\}) \wedge \\
&\quad \text{MEANS-FOR}(\beta,\varphi) \wedge \text{GOAL}(a,\beta)])\} \wedge \\
&\quad \exists G \subseteq T\text{BEL}(a, \text{C-CAN}_G(\varphi) \wedge \\
&\quad \text{STRAT}(\varphi, G) \wedge \\
&\quad \forall i \in G \, willing(i,\varphi))
\end{aligned}$$

Next, the ingredients occurring in the above definition of POTCOOP(a, φ) will be refined. The informational attitude KNOW(a, φ) is discussed in section 2. The predicate MEANS-FOR(β, φ) is meant to associate appropriate means, a possibly complex action β, to a goal φ:

$$\begin{aligned}
&\text{MEANS-FOR}(\beta,\varphi) \leftrightarrow \\
&\quad inevitable(happens(\beta) \rightarrow happens(\beta;\varphi?)).
\end{aligned}$$

Thus MEANS-FOR(β, φ) means that on all paths, after action β happens, the test for proposition φ is successful, i.e. φ is true. Here, the straightforward interpretation of *happens*(β) is that action β happens next on the current path.

We find that an agent a can individually achieve goal φ if there is a (possibly complex) complex action β such that a knows that a itself is the single agent required to perform β, and that β is a means to realize φ. Using the previously defined predicates, individual ability can be formally defined as follows:

$$\begin{aligned}
&\text{CAN}(a,\varphi) \leftrightarrow \\
&\quad \exists\beta(\text{KNOW}(a, \text{AGENTS}(\beta,\{a\}) \wedge \\
&\quad \text{MEANS-FOR}(\beta,\varphi))).
\end{aligned}$$

Here, the primitive predicate AGENTS(β, G), meaning that group G is precisely the set of agents required to perform action β, is taken from [31]. Note that our definition of individual ability itself is not recursive, and differs from the one in [31].

The definition of collective ability of a group G to achieve φ should be decomposed in terms of individual abilities of team members. For this purpose the overall goal φ is split into a number of subgoals $\varphi_1, \dots, \varphi_n$. These subgoals can be viewed as instrumental to the overall goal. Together they *constitute* φ and are compared with the individual abilities of the agents. We propose that a goup G can collectively achieve goal φ if φ can be split into subtasks $\varphi_1, \dots, \varphi_n$ that constitute it, and such that for all these subtasks there is a team member j in G who can individually achieve that subtask. Formally:

$$\begin{aligned}
&\text{C-CAN}_G(\varphi) \leftrightarrow \exists\varphi_1,\dots,\exists\varphi_n \\
&\quad (constitute(<\varphi_1,\dots,\varphi_n>,\varphi) \wedge \\
&\quad \forall i \leq n \exists j \in G\,\text{CAN}(j,\varphi_i)).
\end{aligned}$$

The willingness *willing*(b, φ) to participate in the team formation is modelled as the agent's belief that it is optional that it is possible that it will have the individual intention to reach the overall goal. Formally *willing*(b, φ) stands for BEL$(b, optional \Diamond \text{INT}(b, \varphi))$. This does not conflict with the agent having different intentions INT(b, ψ), even if ψ is inconsistent with the overall goal φ. Note that the agent's willingness to participate in team formation does not mean that the agent will necessarily be a member of the resulting team.

Unfortunately, STRAT(φ, G), the right distribution of commitment strategies, can not be defined in terms of individual commitment strategies. The reason is that this notion is not compositional. In fact we have to take the overall distribution of commitment strategies as primitive and regard it as a kind of constraint on individual commitments. Whenever the set of individual commitment strategies of the members is a suitable distribution for the whole group then the individual commitment strategies are also suitable.

Let $S = \{blind, single\text{-}minded, open\text{-}minded\}$ be the set of commitment strategies and $s_i \in S$ and *constitute*$(< \varphi_1, \ldots, \varphi_n >, \varphi)$ and $j \leq n$ and $i \in G$ then $\mathrm{indstrat}(i, s_i, \varphi_j, G, \varphi)$ denotes the fact that agent i uses commitment strategy s_i to achieve φ_j, as part of the overall goal φ and being a member of the team G.

A suitable commitment strategy for an agent i is defined as follows:

If

constitute$(< \varphi_1, \ldots, \varphi_n >, \varphi) \wedge$
$\bigwedge_{i \in G} \mathrm{indstrat}(i, s_i, \varphi_j, G, \varphi)$
$\rightarrow \mathrm{STRAT}(\varphi, G)$

then

for all i, s_i is a suitable commitment strategy for agent i (to achieve φ_j in team G as part of the overall goal φ).

As output of a successful outcome of this stage, track is kept of the sequence $(G_1, \ldots, G_n)$ of all relevant groups $G \subseteq T$ for which a sees potential for cooperation with respect to the main goal. It will be of use when the need for reconfiguration appears.

Level 2: team formation

Suppose that agent a sees the potential for cooperation to achieve φ. Somewhat different from [31], we find that during the team formation stage agent a attempts to bring it about in some group G that the group has the *collective intention* to make φ true (see section 5.1). The input of this stage is agent a, a formula φ and sequence of potential groups $(G_1, \ldots, G_n)$ as output by stage 1. The output of a successful outcome of this stage is one group G from $(G_1, \ldots, G_n)$ together with a collective intention among G to achieve φ, which includes corresponding individual intentions of all group members. This is done by subsequently attempting to establish the collective intention among G_1, G_2 etc., until this succeeds for some G_i. The sequence of still untried potential groups $(G_{i+1}, \ldots, G_n)$ is stored for revision purposes.

Level 3: plan generation

The input of this stage is a team G together with its collective intention to achieve a goal φ. The final successful outcome of this stage is the collective commitment $\text{C-COMM}_{G,P}(\varphi)$ of the group G based on the social plan P (see section 5.2). In AI and MAS literature many methods of planning have been proposed. From our perspective, the most interesting one is *planning from first principles.* In the MAS context, we may assume that this method includes *task division* and *task allocation.* We will discuss this method now.

During a successful run of the plan generation stage, first, an adequate task division of φ into a sequence of (possibly complex) subtasks $\varphi_1, \ldots, \varphi_n$ is constructed, ensuring the certain realization of the collective goal. These subgoals can be viewed as instrumental to the overall goal. Together they *constitute* φ and are compared with the individual capabilities and opportunities that the agents are believed to have.

This step is followed by *task allocation,* in which actions are associated to the subtasks constructed in task division, and these are given to team members. This results in pairs $< \alpha_i, b >$ of a possibly complex action α_i that realizes task φ_i and an agent b. For an appropriate allocation of actions to agents, not only agents' abilities and resources, but also their commitment strategies are taken into account. To make a social plan complete, the temporal structure among the pairs $< \alpha_i, b >$ should be established. This is reflected in the recursive definition of a *social plan expression.* The plans on which collective commitments are based are always represented as social plan expressions.

Note that team members' characteristics, such as abilities, commitment strategies and resources, may already play a role at the stage of task division. Therefore, meta-knowledge about all involved aspects within the team should be accessible. The result of the planning process is a plan P, represented as a social plan expression. Note that there is a close interplay between task division and task allocation.

In effect, at the plan generation level all agents from a group socially commit to carry out their respective subtasks, and to communicate about these social commitments in order to establish pairwise mutual beliefs about them. This information exchange concludes the *collective* part of plan generation. The final successful outcome of this stage is collective commitment C-COMM$_{G,P}(\varphi)$ of the group G based on the social plan P. Because of our strong notion of collective commitment, this definition of the successful outcome of the stage of plan generation is somewhat stronger than the one in [31].

Level 4: plan execution / team action

This level includes *execution of actions* and the *reconfiguration procedure.* At the level of plan execution, all team members aim at realizing their own subtasks from the collective commitment C-COMM$_{G,P}(\varphi)$ constructed at stage 3. Thus they start executing adequate agent-specific actions. Many different situations may occur, some of which imply reconfiguration among the group, an aspect which is not treated explicitly in [31]. We will discuss reconfiguration in the sequel. In terms of motivational attitudes, plan realization amounts to the maintenance of *social commitments* and associated *individual intentions.*
The successful outcome of the stage of plan execution is that all subtasks making up the social plan P have been carried out by the agents who were socially committed to do them, the team's collective commitment and that by the success of their actions the goal φ of the collective commitment has been achieved.
Now we are ready to define success and failure of the system as a whole. The *system fails* if the overall goal φ is not realizable by any team from the relevant set T; the *system succeeds* otherwise.

6.1 The structure of the system

The diversity of the system's functions should be reflected in the structure of the system: in addition to agents realizing the domain problem solving there need to be agents aiming at the proper *organization* of DPS, all of them operating on different levels. The goals of such organizing agents could be the following:

- To organize the control on particular levels; let us note that here such complex tasks like cooperation, communication and negotiation are involved.

- To realize the reconfiguration method as the interplay between different levels.

7 The reconfiguration method

The dynamics of the system will be reflected in the *cycle of the system.* In order to achieve a given overall goal, one cycle realizes the following consecutive steps: potential recognition, team formation, plan generation, and plan execution.

In the perfect case a group achieves the goal in the way it was planned from the very beginning. In the more common non-perfect case disturbances appear in the first cycle of the system at some level. In such a situation some kind of *reconfiguration* is necessary. Because the need to reconfigure may appear at every moment, the cycle of the system will be referred to as the *reconfiguration algorithm.* The main purpose of the algorithm is *the proper maintenance of collective commitments.* To properly deal with the variety of situations the reasons of disturbances have to be recognized. Therefore, a few definitions will be introduced.

An action execution fails for an *objective reason* if it is not realizable by anybody in the present team in the current state of the world.

An action execution fails for a *subjective reason* if the agent to whom it is delegated does not believe that he can achieve it.

The problem of choosing adequate properties of a reconfiguration method may be viewed as an open question, possibly related to the type of application considered. When formulating the reconfiguration algorithm, we chose some intuitive properties corresponding to classical strategies adopted in backtracking. We postulate that the system behaviour should preserve *continuity.* This means that, if an obstacle appears, the problems are solved by moving up in the hierarchy, but as little as possible. Thus, one moves to the nearest point up in the hierarchy of levels where a different choice is possible. If such a point does not exist anymore, the reconfiguration algorithm fails. In other words, depth-first search is used.

7.1 Conservativity

The continuity criterion is context-independent. As the basis of local decisions a context-sensitive criterion would be valuable. When new rounds at particular levels are necessary, the question arises which results should be preferred based on the failed ones. The answer to this question is certainly context-dependent and requires formulation of a specific notion of the distance between teams, goals, plans, etc. It seems that for a wide class of application domains, the system should behave in a rather *conservative* way.

In our case, conservativity (or *inertia*) entails that the collective commitment that is being carried out should change as little as possible. In particular, conservativity includes:

- When *team formation* is not realizable for the current sequence of teams, a new round of potential recognition is needed, taking care that the new sequence of potential teams is as close as possible to the previous one.

- If a *task division* is not realizable by the current team a round of team formation is needed in order to create a new team for the current collective intention, taking care that as few as possible team members are replaced by new ones.

- When making a new *task division* for the current collective intention and the team, one should take care that as few as possible tasks are replaced by new ones. In particular, one should try to reinclude the tasks that have been carried out already into the new plan, so that hard work is not thrown away unnecessarily.

- When making a new *task allocation* for the current task division one must take care that as few as possible tasks (and only subjectively failed ones) are reassigned.

- During replanning as a whole, one should take care that the new social plan is as close as possible to the previous one, taking into account tasks that have already been successfully achieved. This entails that even in the case of a new task division, one should remember the previous task allocation and give "old" (or similar) tasks to agents that were assigned them before. Thus, division and allocation should not be viewed independently.

7.2 Reconfiguration algorithm

The (informal) description of the reconfiguration algorithm below is meant to be generic: a pattern of behavior is described in terms of abstract level-associated procedures (i.e. *potential-recognition, team-formation, task-division, task-allocation, plan-execution*), without fixing any particular method or strategy. Input and output parameters, as well some other conditions of these complex procedures are commented in the algorithm itself. Let us stress, that each of these procedures may succeed or fail - in this sense the structure of each abstract level is analogical.

To make the algorithm's structure more transparent, we decided to bind a label with each level-associated procedure and to use the *goto* statement.

In the algorithm an assignment of new social and individual motivational attitudes is made to create a new collective commitment. In this sense this algorithm may be viewed as a *revision* of motivational attitudes. In the body of the algorithm phases of belief revision and motivational attitudes revison are distinguished, without further refinement. They are realized by abstract procedures *belief-revision* and *motivational-attitudes-revision*, after *motivational-attitudes-assignment*. Let us note however, that in the description of the algorithm we have resigned from explicitly splitting collective motivational attitudes into individual and social ones. Also, for the sake of simplicity, the above procedures haven't fixed the number of parameters, but their proper use is always implied by the context.

Let us stress that system failure and success are realized by the complex procedures *system-failure* and *system-success*, respectively. It is also assumed that all the required information is available at the development-time.

Reconfiguration algorithm:

```
begin
{input of the system:
Q - a goal of agent S;
S - the agent that is input to the system and that will recognize potential;
T - a set of agents from whom potential teams are selected}
A: potential-recognition (Q, S, H, T);
   {input: Q, S, T}
   if not (potential-recognition-succeeded) then
      {S does not see any potential for cooperation with respect to goal Q}
      system-failure;
      STOP
   else
   {potential recognition succeeded - output: H - a sequence of teams H =($G_1, \ldots, G_n$)
    with the potential to realize Q}
   initialization-of-motivational-attitudes(Q,S);
   {the attitude POTCOOP($S, Q$) is established}
B: team-formation (Q, S, H, G);
   {input: Q, S, H}
   if not (team-formation-succeeded) then
   {the collective intention w.r.t. Q cannot be established among any of the teams from H;
    return to the potential recognition level for S to construct a new sequence of potential
    teams for which it sees cooperation potential w.r.t. Q}
      goto A
   else
   {team formation succeeded - output G: a set of agents aiming to realize Q; Suppose G is the
     first unused $G_i$ from H for which the collective intention towards Q can be established}
   motivational-attitudes-assignment (Q, G);
   {the collective intention C-INT$_G(Q)$ of G towards Q is established here}
C: task-division (Q, G, R);
   {input: Q, G}
   if not (task-division-succeeded) then
   {task division failed; return to the team-formation level in order to attempt the first unused
    $G_{i+1}$ from H to establish collective intention towards Q}
   goto B
   fi;
   {task division succeeded - output: R - a sequence of subtasks together realizing Q;
    i.e. the first part of a social plan P}
D: task-allocation (Q, G, R, P);
   {input: Q, G, R}
   if not (task-allocation-succeeded) then
   {task allocation failed; return to the task division level}
     goto C
   fi;
   {task allocation succeeded - output: a social plan P corresponding to subtask sequence R}
      motivational-attitude-assignment (Q, G, R, P);
      {the collective commitment C-COMM$_{G,P}(Q)$ among G to goal Q with respect to social
        plan P is established (including corresponding social commitments to subtasks from R)}
      {plan execution part starts here}
   E: plan-execution (Q, G, R, P);
```

```
        {input Q, G, R, P}
        if plan-execution-succeeded then
        {all actions that constitute the plan P are successfully executed; agents' beliefs and
         motivational attitudes need to be revised to reflect that the overall goal Q is achieved
         as well as agents' subgoals from R}
           belief-revision (Q, G, R);
           motivational-attitudes-revision (Q, G, R, P);
           system-success;
           STOP
        elseif
           {some action execution failed: differentiation of reasons for failure}
           F1:if subjective-reason-for-failure(Q, P, G, R, R1, R2) and
           not (objective-reason-for-failure(Q, P, G, R, R1 ,R2)) then
           {R1: the sequence of tasks from R that have been successfully achieved thus far;
            R2: the sequence of pairs (A, X) of tasks that failed plus reasons for failure,
            where X=ob or X=sub}
           {for every action that failed, it failed for a subjective reason, i.e.
           $\forall A \in R\, \forall X((A, X) \in R2 \rightarrow X = sub)$}
            belief-revision (G, R, R1, R2));
            if task-reallocation-possible then
            {for every action that failed for a subjective reason, there is another team
             member believing it can achieve it, i.e. $\forall A \in R\, \forall M_1, M_2 \in G((A, sub) \in R2$
             $\wedge \mathrm{COMM}(M_1, M_2, A) \rightarrow \exists M_3 \in G(M_1 \neq M_3 \wedge \mathrm{BEL}(M_3, \mathrm{CAN}(M_3, A))))$;
             before an attempt at task reallocation based on P, R1 and R2 is made,
             a revision of motivational attitudes is needed}
                 motivational-attitudes-revision (Q, P, G, R);
                 goto D
              elseif
              {no task reallocation is possible: a new task division is needed}
               goto C
           fi
        fi
        {there are also objective reasons for failure, i.e. $\exists A \in R((A, ob) \in R2)$}
        F2:  belief-revision (G, R, R1, R2);
        {investigation whether the objective reason for failure blocks achieving the goal Q}
        if blocked(Q) then
        {the objective reason for failure blocks achieving the goal Q}
           system failure;
           STOP
        else
        {the objective reason for failure does not block achieving the goal Q;
         a new task division is needed, return to the task division level}
           goto C
        fi
     fi
  end
```

Let us trace the cycle of the system during a realization of the reconfiguration algorithm. When consecutive steps of reconfiguration go well, changes of individual, social and collective motivational attitudes are commented in the description of the algorithm. Let us focus on the points where failure of the four main stages of the process takes place.

The failure of the potential recognition level (see the label A) meaning that agent S does not see any potential for cooperation with respect to the goal Q, leads to the total failure of the system.

The failure of the team formation level (see the label B), meaning that the collective intention with respect to Q cannot be established among any of the teams from H, requires a return to the potential recognition stage to construct a new sequence of potential teams.

The failure of the task division level (see the label C) requires a return to the team formation level in order to establish a collective intention in the chosen new team from H and may be viewed as the reconfiguration of the team together with a change of motivational attitudes on the level of collective intention and respective individual attitudes.

The failure of the task allocation level (see the label D) requires a return to the task division level in order to create a new sequence of tasks. This may be viewed as the reconfiguration of the first part of a social plan P to realize the goal Q.

When, finally, a collective commitment is established, the failure of some actions together constituting the social plan P requires the reconfiguration of motivational attitudes on the level of collective commitment.

7.3 Complexity of the reconfiguration algorithm

Let us recall that the reconfiguration algorithm is an abstract one. Thus, it is meant to be instantiated for each particular application.

When defining the levels we abstracted from particular methods meant to realize level-associated goals, but instead formulated their final results and associated them with appropriate motivational attitudes. Procedures realizing level-associated goals are rather complex.

Especially the plan generation / formation level, including task division and task allocation, is complex from both the AI and the MAS perspective. As for the first perspective, planning is involved. If done from first principles, planning is in general undecidable [4], while for limited domains it may still be tractable. If, on the other hand, a plan library is used, searching the library may be complex as well. As for the MAS perspective, the aspects of communication, negotiation and coordination come to the fore on the level of plan generation. Even though the work on negotiation is still in progress, it is quite clear that the procedures being proposed are usually rather complex [25]. The same holds for coordination.

On the level of plan execution, belief revision, which is known to be NP-hard, is repeatedly performed. Also, if there are objectively failed tasks, it needs to be checked whether their failure blocks the overall goal. This kind of consistency check is co-NP complete. Let us stress that all the complex aspects mentioned here have to be treated in any rigorous approach to the reconfiguration problem.

As for the global structure of the reconfiguration algorithm, it is based on backtracking search. In the generic case, when not using any domain-dependent information, context-dependent improvements as obtained in informed search methods cannot be made.

However we did use forward checking in the algorithm in the check whether an objectively failed task blocks the overall goal. Next, in order to be certain that the search method is complete and optimal, iterative deepening may be used. It doesn't increase the time or space complexity when compared with depth-first search. Due to its exponential space complexity, breadth-first search is no option for reconfiguration.

We would like to stress that our definition of collective commitment ensures efficiency of reconfiguration in two ways. Firstly, all the motivational attitudes occuring in the definition of collective commitment are defined in a non-recursive way. This allows straightforward revision of all relevant motivational attitudes. Secondly, only pairwise social commitments to subtasks appear in the definition of collective commitment, so that replanning and motivational attitudes revision are made less complex. Thus, in contast to [31], using our definition of collective commitment it is often sufficient to revise some of the pairwise commitments, instead of involving the entire team in the replanning process.

8 Discussion and conclusions

In this paper we have provided a formalization of social and collective motivational attitudes of agents within strictly cooperative groups. Special attention was paid to the strongest attitude: collective commitment. Our approach is inspired by Castelfranchi's pre-formal discussion [2]. We share his opinion that commitments are stronger than intentions, in the sense that it is commitments, not intentions, that trigger action execution. Our contribution is the formal characterization of collective commitments, viewed from the perspective of teamwork, which is reflected in a plan towards achieving a given goal. The collective commitment hinges on pairwise social commitments between the team members to realize the actions resulting from the generated decomposition of the overall goal.

Then, we have provided an abstract multilevel architecture which makes it possible to properly deal with CPS, based on the BDI paradigm. In particular, we stressed the need of the proper treatment of all motivational attitudes — individual, social, and collective, through all stages of CPS. To achieve this we introduced, as in [31], levels of abstraction reflecting the main aspects of CPS, namely potential recognition, team formation, plan generation, and plan execution. The main contribution is the reconfiguration algorithm respecting the property of continuity, which entails that replanning should be more efficient than when other (e.g. blind) methods are used. Also our definition of collective commitment is meant to ensure efficient replanning because of the pairwise social commitments involved. In addition, it seems that in many domains, conservativity is the appropriate criterion to guide local choices during reconfiguration.

When defining a general-purpose reconfiguration method some problems have been left aside. We do not take into account questions concerning heterogeneity of the system. Our agents are allowed to be dissimilar and to have different problem solving perspectives, reflected in local expertise of domain solvers and global / strategic expertise of CPS organizers. However, all of them need to communicate, and some to cooperate and coordinate, despite their differences. We agree with [30] that agents' diversity should be restricted by possessing common ontologies, in this case with respect to action effects and motivational attitudes.

Tuomela Tuomela95 considers we-intentions Instead of collective commitments to be the prerequisite for collective activity. His pre-formal definition is agreement-based, in contrast to our plan-based collective commitments. In his view, this agreement need not be fully specified. Therefore, there is no concrete representation of collective plans leading to collective actions in his approach.

In [16] Grosz and Kraus give an interesting definition of partial shared plans, but do not treat reconfiguration as a separate subject. Rao, Georgeff and Sonenberg [23] consider related issues with an emphasis on the ontology and semantics of social agents carrying out social plans, but they do not provide a generic reconfiguration method either and work with pre-given plans instead of plans that can be adapted on the fly. Also, their definition of collective intention is not sufficiently strong, as we argued in subsection 5.1.

The need for reconfiguration was recognized by other authors, and some of the abovementioned papers treat certain aspects of reconfiguration or apply it to specific cases. Our paper, in contrast, provides a rather methodological approach to maintaining all motivational attitudes.

Haddadi [17] gives an internal or prescriptive approach that characterizes the stages of CPS in a manner similar to [31]. She introduces the notions of pre-commitments and commitments between pairs of agents and presents an extensive and well-founded discussion of their properties, including important aspects like communication. However, in contrast to our approach, she does not go beyond the level of pairwise commitments and is not explicit about their contribution to collective behavior in a bigger team.

Another question addressed in this paper is when team members can responsibly drop their social commitments. We adopt the perspective that strategies for dropping a social commitment are agent related. This is in contrast to Wooldridge and Jennings' social conventions for adopting and dropping joint commitments: their conventions are task related. When defining the commitment strategies in the context of teamwork, we introduced the aspects of communication and coordination with interested team members.

9 Further research

The reconfiguration algorithm is meant to be generic: a pattern of behavior is described in terms of abstract level-associated procedures, viewed as a sort of black box with specified input and output parameters. Future work will be to formalize the framework and to prove desired properties like efficiency of replanning. Then, the reconfiguration algorithm may be implemented for some applications.

The next step should be a transformation of *black* boxes into *glass* ones. When designing multiagent systems a good balance between communication and reasoning is of importance, nevertheless communication is always necessary. In recent systems models of communication range from rather inflexible communication protocols to more sophisticated constructions based on speech acts. What is missing is a more in-depth analysis of different types of communication. In [9] we have tried at least partly to fill this gap. Our approach is based on the rigorous typology of dialogues, distinguishing *persuasion*, *negotiation*, *inquiry*, *deliberation*, and *information seeking*. In that paper we give a short characterization of the above dialogue types and discuss their role during teamwork, on the basis of the four-level structure discussed in the present paper. In [5] the analysis for the first two stages of CPS, potential recognition and team formation, is taken a step further to the level of speech acts. Future work will be to complete the formal normative model of CPS related dialogue types. Thus it will be possible to prove that in given circumstances the dialogue results in a certain outcome.

10 Acknowledgements

We would like to thank Gerard Renardel de Lavalette and Theo Kuipers for their helpful comments.

References

[1] M. Bratman. *Intension, Plans, and Practical Reason.* Harvard University Press, Cambridge (MA), 1987.

[2] C. Castelfranchi. Commitments: From individual intentions to groups and organizations. In Lesser [20], pages 41–48.

[3] L. Cavedon, A. Rao, and G. Tidhar. Social and individual commitment (preliminary report). In L. Cavedon, A. Rao, and W. Wobcke, editors, *Intelligent Agent Systems: Theoretical and Practical Issues*, volume 1209 of *Lecture Notes in Artificial Intelligence*, pages 152–163. Springer Verlag, Berlin, 1997.

[4] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[5] F. Dignum, B. Dunin-Kęplicz, and R. Verbrugge. Dialogue in team formation: A formal approach. Technical report, submitted, 1999.

[6] B. Dunin-Kęplicz and A. Radzikowska. Actions with typical effects: Epistemic characterization of scenarios. In Lesser [20], page 445.

[7] B. Dunin-Kęplicz and A. Radzikowska. Epistemic approach to actions with typical effects. In *Proceedings ECSQARU'95*, pages 180–189, Fribourg, 1995.

[8] B. Dunin-Kęplicz and A. Radzikowska. Modelling nondeterminstic actions with typical effects. In *Proceedings DIMAS'95*, pages 158–166, Cracow, 1995.

[9] B. Dunin-Kęplicz and R. Verbrugge. Technical report.

[10] B. Dunin-Kęplicz and R. Verbrugge. Collective commitments. In M. Tokoro, editor, *Proceedings Second International Conference on Multi-Agent Systems*, pages 56–63, Menlo Park (CA), 1996. AAAI-Press.

[11] B. Dunin-Kęplicz and R. Verbrugge. A methodology for maintaining collective motivational attitudes during teamwork. In F. Garijo and Ch. Lemaitre, editors, *Multi Agent Systems Models Architectures and Applications: Proceedings of the Second Iberoamerican Workshop on DAI and MAS, October 1-2 1998*, pages 45–60, Toledo, Spain, 1998.

[12] B. Dunin-Kęplicz and R. Verbrugge. A reconfiguration algorithm for the maintenance of collective commitments. In Y. Demazeau, editor, *Proceedings Third International Conference on Multi-Agent Systems*, pages 421–422, Los Alamitos (CA), 1998. IEEE Computer Society.

[13] B. Dunin-Kęplicz and R. Verbrugge. A reconfiguration algorithm for distributed problem solving. Technical report, submitted, 1999.

[14] A.E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier and MIT-Press, Amsterdam and Cambridge (MA), 1990.

[15] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, MA, 1995.

[16] B.J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[17] A. Haddadi. *Communication and Cooperation in Agent Systems: A Pragmatic Theory*, volume 1056 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, Berlin, 1995.

[18] J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37:549–587, 1990.

[19] N. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *Knowledge Engineering Review*, 3:223–250, 1993.

[20] V. Lesser, editor. *Proceedings First International Conference on Multi-Agent Systems*, San Francisco, 1995. AAAI-Press and MIT Press.

[21] H.J. Levesque, P.R. Cohen, and J.H.T. Nunes. On acting together. In *Proceedings Eighth National Conference on AI (AAAI90)*, pages 94–99, Menlo Park (CA), Cambridge (MA), 1990. AAAI-Press and MIT Press.

[22] A. Rao and M. Georgeff. Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of the Second Conference on Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufman, 1991.

[23] A. Rao, M. Georgeff, and E. Sonenberg. Social plans: A preliminary report. In E. Werner and Y. Demazeau, editors, *Decentralized A.I.-3*, pages 57–76, Amsterdam, 1992. Elsevier.

[24] A.S. Rao and M.P. Georgeff. Formal models and decision procedures for multi-agent systems. Technical Report Technical Note 61, Australian Artificial Intelligence Institute, Carlton, Victoria, 1995.

[25] J.S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT-Press, Cambridge (MA), 1994.

[26] K. Segerberg. Bringing it about. *Journal of Philosophical Logic*, 18:327–347, 1989.

[27] M. Singh. Commitments among autonomous agents in information-rich environments. In M. Boman and W. Van de Velde, editors, *Multi-Agent Rationality (Proceedings of MAAMAW'97)*, volume 1237 of *Lecture Notes in Artificial Intelligence*, pages 141–155. Springer Verlag, Berlin, 1997.

[28] R. Tuomela. *The Importance of Us: A Philosophical Study of Basic Social Notions*. Stanford Series in Philosophy. Stanford University Press, Stanford (CA), 1995.

[29] R. Tuomela and K. Miller. We-intentions. *Philosophical Studies*, 53:367–390, 1988.

[30] R. Weihmayer, R. Brandau, and Hong Shinn. Modes of diversity: Issues in cooperation among dissimilar agents. In *Proceedings 10th International Workshop on Distributed AI (AAAI/MCC)*, Bandera, Texas, October 1990.

[31] M. Wooldridge and N.R. Jennings. Towards a theory of collective problem solving. In J.W. Perram and J.P. Muller, editors, *Distributed Software Agents and Applications*, volume 1069 of *Lecture Notes in Artificial Intelligence*, pages 40–53. Springer Verlag, Berlin, 1996.

XJ DOME — AN ENVIRONMENT FOR THE DEVELOPMENT AND USAGE OF MOBILE AGENTS.

Yuri G. Karpov, Andrei V. Borshchev, Alex E. Filippoff and Kirill R. Bolshakov

Experimental Object Technologies (XJ) & St. Petersburg Technical University, Russia
dome@xjtek.com *http://www.xjtek.com*

Abstract

XJ DOME is a set of tools and techniques for those who wish to speed up development of Distributed COM applications and improve their quality. DOME supports graphical modeling, code generation, simulation, deployment, monitoring and management. The simulation mode enables the developer to simulate the entire distributed application in virtual time on a single machine. After simulation step the application can be deployed onto the target network and managed via DOME Application Viewer. During run-time DOME platform enables the developer to collect and watch statistics, inspect threads and synchronization objects, view logs. DOME platform supports building of mobile agent systems on top of DCOM services. It provides for agent migration and DCOM security.

Keywords: *multi-agent systems, DCOM, simulation, modeling, RAD*

1. Introduction

Mobile agents are attracting interest from fields of distributed systems, information retrieval, electronic commerce and artificial intelligence as a rapidly evolving technology. This area suffers a lack of industrial-strength development tools support.

In this paper we present XJ DOME — run-time and development environment for building mobile agent system on top of Microsoft DCOM services. It also provides graphical specification of object behavior, as well as simulation and visualization services. XJ DOME may tightly integrate with MS Visual C++ Developer Studio. Thus, DOME provides the developer with friendly environment from specification through debugging to real execution stage.

2. The current state of mobile agent systems market

The vast majority of the agent systems available are research prototypes and only a few of them have users outside their own university or research institute. The most known are Aglets (IBM, Japan), Mole (University of Stuttgart, Germany), Telescript (General Magic, USA) and AgentTcl (Dartmouth College, USA). Aglets and Mole support Java, AgentTcl supports Tcl, Telescript has its own language.

However, there are no high-level rapid application development (RAD) environments for agents development. Also, at the moment there is no mobile agent system based on DCOM platform. The implementation of mobile agent system for DCOM enables to utilize its performance advantages (execution of native code and full access to OS services) and integration with MS Windows NT security.

3. XJ DOME run-time environment

XJ DOME run-time environment supports the following models of mobile agents:

- Lifecycle
- Computational
- Communication
- Navigation
- Security

The lifecycle model provides services to create, destroy, save and restore mobile agents. The computational model heavily relies on Win32 services at the moment. The navigation model handles all issues referring to transporting an agent between two places. The communication model defines communication between agents. The security model defines rules of mutual access for agents and network.

We build our communication, navigation and security models on top of Microsoft DCOM. This allows the developer to fully exploit the advantages of Microsoft industry standard for Windows environments. This also greatly simplifies dealing with numerous security issues intrinsic to mobile agent systems.

4. DOME Application Editor

The basic services provided by XJ DOME Developer Studio are:

- Creation of COM components in visual environment with complete code generation (for both static and dynamic components)
- Debugging of timings and synchronization and performance estimation of a distributed application by simulating it in virtual time on the developer's machine
- Deployment of the application components over the target network
- Monitoring of the distributed application: collect and display statistics, inspect threads and synchronization objects, view logs, etc.
- Management of the distributed application via COM interfaces using standard controls

For rapid prototyping of distributed COM applications XJ DOME offers Application Editor, see Figure 1. It includes graphical COM Object Diagram and Statecharts editors. DOME Application Editor generates the complete application code including IDL, C++, resources, and MS Visual C++ project. It builds the application components using MS Visual C++ command-line compiler. DOME Application Editor drastically saves developer's time on the early design stages. Later on, when "COM skeleton" of the application becomes more stable, the developer can continue with MS Visual C++ environment using built-in DOME Wizards.

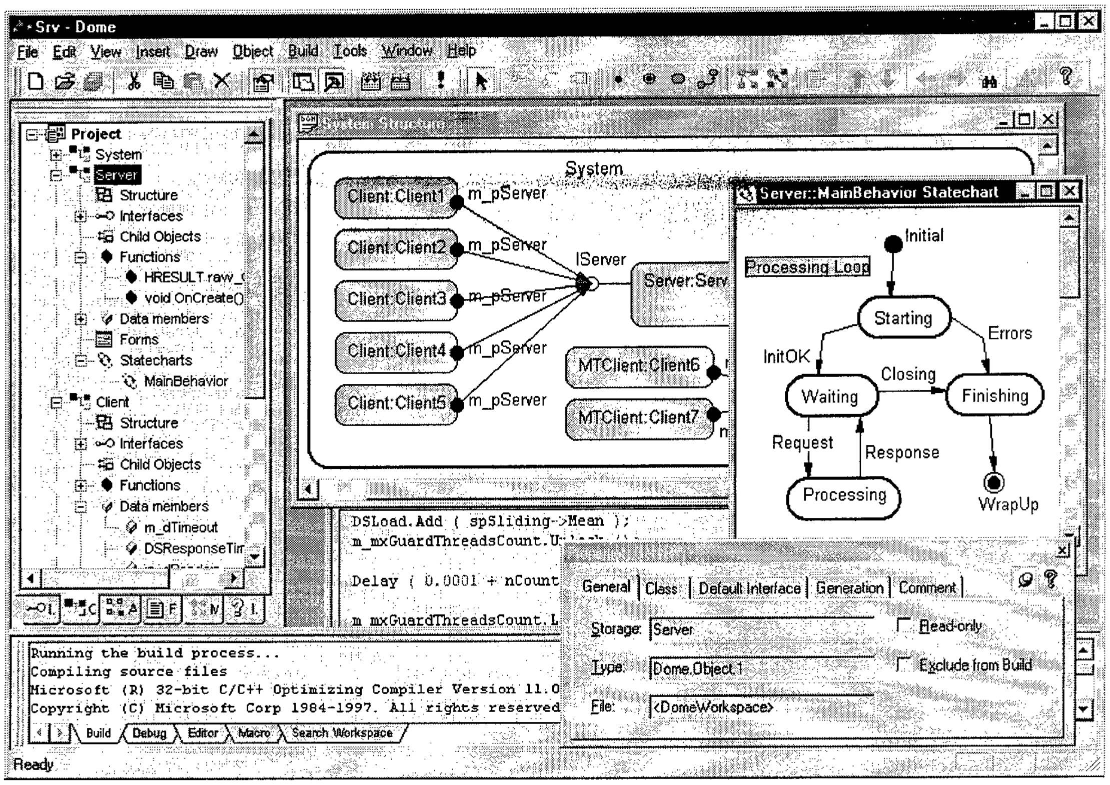
5. Wizards for MS Visual C++

DOME can be used with new projects as well as with the legacy software. In the latter case DOME functionality can be added to the application gradually. By following a set of simple rules one can easily instrument COM objects under development to use DOME simulation, deployment, visualization and management facilities. To automate this work DOME adds to MS Visual C++ development environment a set of wizards covering all related tasks:

- DOME App Wizard — creates a skeleton application project.
- Add DOME Object — adds an object with support of standard DOME interfaces and skeleton implementation to the application project.
- Add Child Object — adds aggregated or contained child object to DOME object.
- Add COM pointer — adds a COM pointer to DOME object. DOME Objects communicate by calling methods of each other via COM pointers that are in turn set up by DOME run-time environment.
- Add DOME Thread — adds a control thread to DOME object. DOME objects can have several control threads and spawn them dynamically.
- Add DOME Statistics — adds an ability to collect statistics to DOME object. On execution stage the statistics is available for monitoring in DOME Viewer.
- Add DOME Log — adds a log access point to DOME object. DOME a feature that allows the user to watch logs of several objects deployed to different hosts.

6. Simulation

Simulation is used for preliminary analysis of the application correctness and estimation of its performance. In the simulation mode the most detailed information on threads and synchronization objects is available online in DOME Application Viewer, see Figure 2. The user can run the application step-by-step, stop upon a certain condition, e.g. when enough statistics is collected, etc. Using automation the developer can program DOME to run multiple simulation sessions to find optimal parameter values or to test the application scalability. All the simulation is performed on a single developer's machine.
DOME simulates the application in virtual time, thus making arbitrary complex experiments possible on a single workstation.
Simulation is supported by DOME Engine that implements IDomeEngineSite interface. The application objects developed according to DOME technology invoke the functions creating new objects, threads, synchronization objects, delaying thread execution, waiting for events, etc. through IDomeEngineSite. In the normal execution mode such call is transparently passed to the local operating system (in fact, the call does not even leave the local machine, as the engine is represented there by a lightweight DOME Engine Proxy object). In the simulation mode, the engine takes care of thread scheduling, synchronization and time.

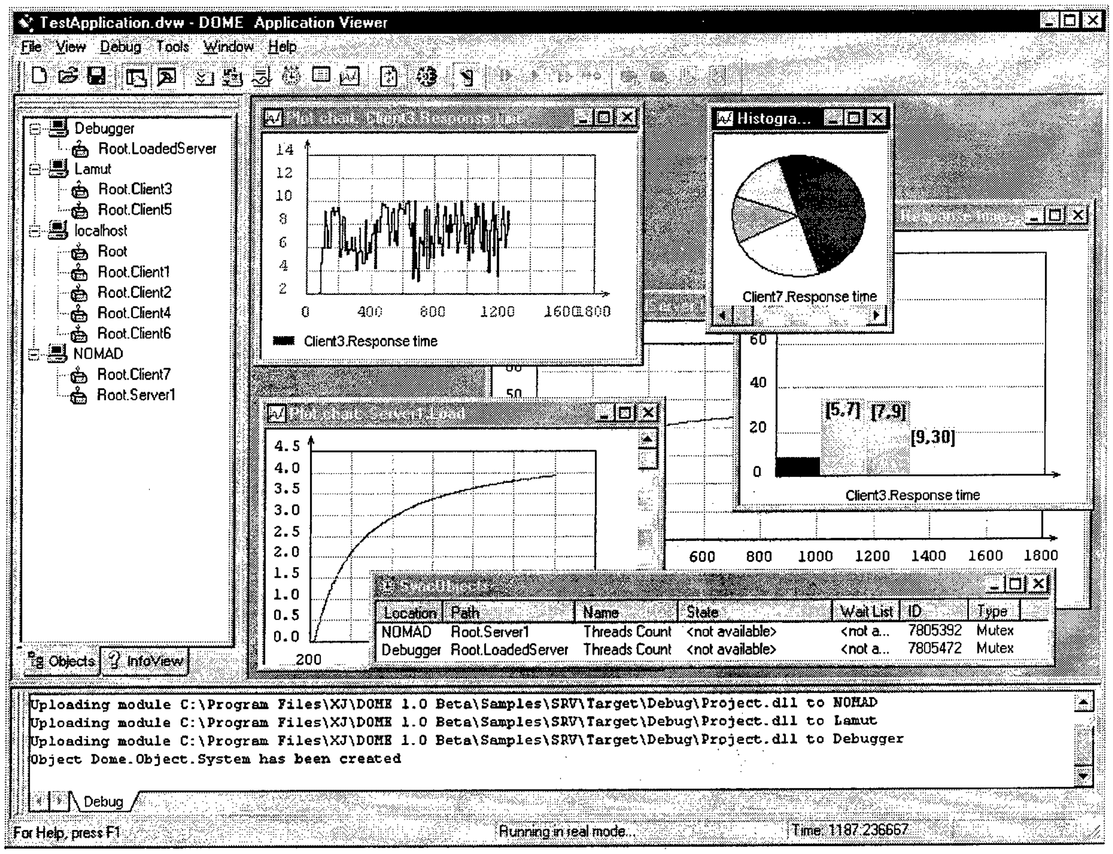


```
DSLoad.Add ( spSliding->Mean );
m_mxGuardThreadsCount.U

Delay ( 0.0001 + nCount

m_mxGuardThreadsCount.L
```

Storage: Server
Type: Dome.Object.1
File: <DomeWorkspace>
Read-only [ ]
Exclude from Build [ ]

```
Running the build process...
Compiling source files
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 11.0
Copyright (C) Microsoft Corp 1984-1997. All rights reserved
```

Figure 1 XJ DOME Application Editor



Location	Path	Name	State	Wait List	ID	Type
NOMAD	Root.Server1	Threads Count	<not available>	<not a...	7805392	Mutex
Debugger	Root.LoadedServer	Threads Count	<not available>	<not a...	7805472	Mutex

```
Uploading module C:\Program Files\XJ\DOME 1.0 Beta\Samples\SRV\Target\Debug\Project.dll to NOMAD
Uploading module C:\Program Files\XJ\DOME 1.0 Beta\Samples\SRV\Target\Debug\Project.dll to Lamut
Uploading module C:\Program Files\XJ\DOME 1.0 Beta\Samples\SRV\Target\Debug\Project.dll to Debugger
Object Dome.Object.System has been created
```

Figure 2 XJ DOME Application Viewer

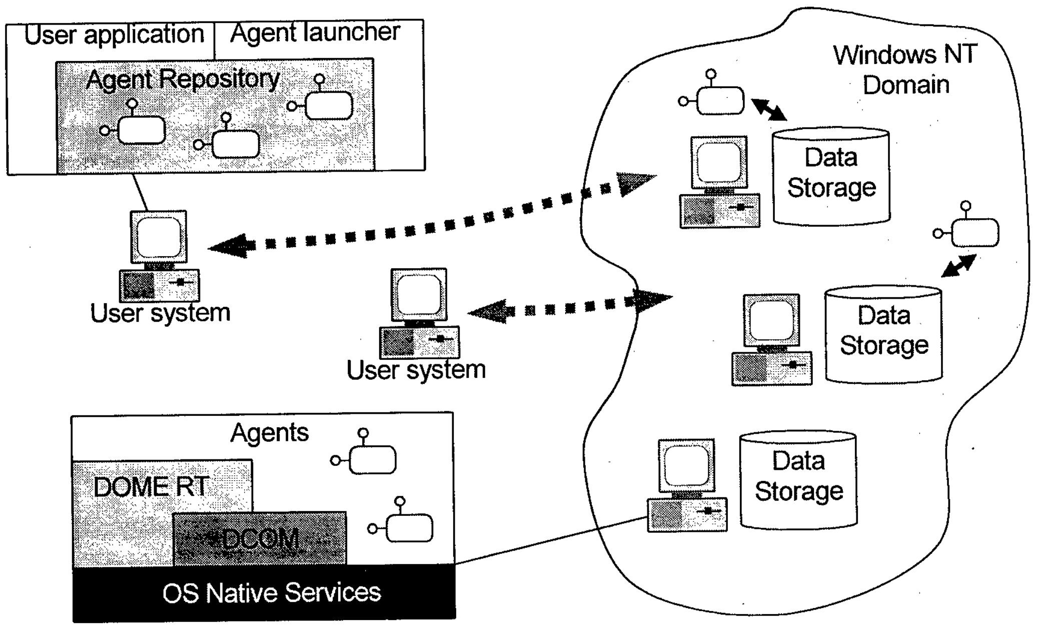


Figure 3 An example of information retrieval system for distributed data storage

7. Visualization

The user can monitor the running application with DOME Application Viewer. The viewer retrieves the information via `IDomeObject` interfaces implemented by DOME-compatible application components and displays the global picture of the application, including:

- Application objects and their hierarchy
- Threads
- Synchronization objects
- Statistics
- Logs
- Inspection views
- `IDispatch` interfaces of objects

The details of the displayed information depend on the execution mode. Namely, in the simulation mode the user can watch the current states of the synchronization objects and threads, and wait queues, whereas in the normal mode these details are not available.

The user can request information about running agents on the specific nodes and view their statistics, as well as access their properties and status information.

8. Management

The user can manage DCOM application with DOME Application Viewer. Before the application starts the user chooses the execution mode (simulation or normal), root objects and gives deployment instructions for static objects.

When the application is running, commands available in the viewer depend on the execution mode. In simulation mode the user has full control over the application execution. Since DOME Engine manages time and synchronization, the user can run the application step-by-step, stop, watch the activity of the selected object, etc. In the normal execution mode time and synchronization are managed by the operating system. In any mode the user can change the COM properties of any application object with DOME IDispatch Browser.

9. Example

As an example (see Figure 3), consider a file searching system. Its purpose is to find a file with a name corresponding to the given pattern and containing given text in it. The search system consists of two parts. The first one is an application that interacts with the user, requests corresponding patterns and reports the result on search completion. Another part of the search system is a mobile agent that travels through the domain and performs search procedure on every host in the domain. Due to the use of DCOM security model the agent has the same privileges as the user who launched it. As the agent finishes visiting hosts in the domain, it returns to the launching system and transfers the results to the front-end application, which reports them to the user.

10. Future Works

As DOME is considered as a framework technology that can be used as a basis for building distributed applications with predictable quality of service, we are working in three directions:

- Implementing general-purpose distributed algorithms in DOME objects, such as distributed termination, distributed snapshot and distributed deadlock detection.
- Developing DOME object-compatible simulation models of communication and navigation models for mobile agent systems and communication media (networks and protocols) for better prediction of application performance.
- Incorporating UML Statecharts engine into DOME objects for enhancing clarity and expressive power of object behavior specification.

Bibliography

1. J. Baumann, F. Hohl, K. Rothermel and M. Strasser. Mole - Concepts of a mobile agent system. World Wide Web, 1 (1998), pp.123-127.
2. D. Krieger and R.M. Adler. The Emergence of Distributed Component Platforms. IEEE Computer, March 1998, pp. 43-53.
3. Andrei V. Borshchev, Yuri G. Karpov and Victor V. Roudakov. Systems Modeling, Simulation and Analysis Using COVERS Active Objects. Proceedings of the IEEE International Conference and Workshop on Engineering of Computer Based Systems, March 24-28, Monterey, California, 1997.
4. Andrei V. Borshchev, Alex E. Filippoff and Yuri G. Karpov. Developing, Simulating and Managing Distributed COM Applications with XJ DOME. In Proceedings of the 1st International Workshop on Computer Science and Information Technologies, Moscow, January 18-22, 1999, Volume 2.

NEGOTIATION ON DATA REALLOCATION IN DISTRIBUTED INFORMATION SYSTEMS

Vladimir V. Mazalov, Vladimir T. Vdovitsyn, Vladimir V. Tarasov

Department of Mathematics and Data Analysis,
Karelian Research Centre of the Russian Academy of Sciences
11 Pushkinskaya St., Petrozavodsk, Karelia, 185610, Russia,
E-mail: {mazalov, vdov, tarasov}@krc.karelia.ru

Abstract

This paper concerns the problem of data reallocation in systems consisting of several informational servers. This problem may emerge when each server processes queries of the clients in its geographical area and it has to often retrieve the needed data from the other servers, that resulting in profit losses. Under these circumstances data reallocation may take place after the negotiation among the servers.

Keywords: *negotiation, distributed information systems, data allocation.*

Nowadays large information systems need to be distributed and working in network environments. Such a system usually consists of a set of database servers located in different geographical areas. Each server contains some information related to specific topics. The pieces of information stored on different servers are not intersected. When a client makes a request, the server either finds the needed data on itself or has to ask another server for the data. As an example we may mention the Data and Information System component of the Earth Observing System of NASA [1]. Another example is the TORIS system [2]. TORIS is a toponimic research system in Northwest Russia and consists of a number of TORIS-servers which contains data on toponims of different language origin and of different categories. For processing a query a server receives a certain amount of payment, but if it has to retrieve some data from the other servers, it pays itself for the data and transportation. As long as the server mainly exploits data stored locally, it has some acceptable profit. But when, at some moment, the situation changes and the server begins to ask for data another servers more and more, it may have substantial losses. So the current data allocation need to be reallocated in order all the servers have profit of some predefined level.

Let us describe the above mentioned situation in a more formal way. Suppose we have a set of information servers distributed in the Internet. Let the set be denoted by $S = \{s_1, s_2, \ldots, s_n\}$, $n > 2$. Each server has a number of datasets stored on it. Let D denote the whole set of datasets: $D = \{d_1, d_2, \ldots, d_m\}$, $m > 1$. Each dataset has the length $l(d_i)$ measured in data units. D_i ($D_i \subseteq D$, $i = 1,\ldots,n$; $\bigcup_i D_i$, $i = 1,\ldots,n$; $\bigcap_{i,j} D_i \neq \varnothing$, $i,j = 1,\ldots,n$, $i \neq j$) is the number of datasets stored on s_i, so we arrive at the distribution of the datasets among the servers: $W = \{D_1, D_2, \ldots, D_n\}$, $n > 2$. Each server s_i has expenses due to the storage of data: $st\_cost_i * l(D_i)$, where st_cost_i is the cost of storing one data unit on a server.

Every client must pay $cost_i$ for processing one query by his server s_i. If the server s_i cannot process the query itself, it asks s^* for the data and has to pay $dist(s^*)*tr\_cost*\ l(d^*)+sr\_cost$, where $dist(s^*)$ – virtual distance to s^*, tr_cost – is the cost of transportation of one data unit over one distance unit, sr_cost – the payment for s^* to process the query from s_i.

Thus, the server s_i has the following profit per a day:

$$P_i = cost_i*K_i - \sum_{d^*} (\ dist(s^*)*tr\_cost*\ l(d^*)+sr\_cost) - st\_cost_i*\ l(D_i),$$

where K_i is the whole number of queries that s_i processed, d^* is the datasets which were retrieved from the other servers and D_i is the set of datasets stored locally.

At every moment $t \in \{1, 2, \dots, T\}$ the servers calculate their profits (expenses), rates of requesting both their own (local) datasets and remote datasets. And if the profit of the server s_i become less than the predefined level U_i, then s_i may propose reallocation W^* of the datasets so its profit greater than or equal to U_i and the others reply 'Yes' or 'No'.

Let us consider the possible situations.

1. All the servers said 'Yes'. In this case the proposed reallocation is simply implemented.
2. There is a server that said 'No'. Then we have a set A of severs which agreed with the proposed reallocation and a set B of severs which disagreed with the proposal. There might also be a set C of severs which are indifferent. (If $C = \varnothing$, then $A \cup B = S$).

Under these circumstances the following restrictions take place:

- the servers of A stop answering queries from the servers of B and vice versa;
- as the servers cannot processed some queries they are imposed with penalties so their profit would be $P_i' = P_i - \mu(K')$, where K' is the number of unprocessed queries.

Hence, until the servers reach an agreement on data reallocation, they have losses. In order to stop having losses the server has to negotiate on data reallocation, and as a negotiation protocol we use Nash bargaining solution.

References

1. Schwartz, R., Kraus, S. Negotiation On Data Allocation in Multi-Agent Environments, In *Proc. of the AAAI-97*, pp. 29-35, 1997.
2. Kert, G. M., Vdovitsyn, V. T., Veretin, A. L. Toponimic research system in Northwest Russia: The TORIS system. In *Proc. of the Scient. Conf. "Karelia and Norway: the Main Trends and Prospects of Scientific Cooperation" held in Karelian Research Centre RAS.* – Petrozavodsk, 1997, pp. 104–108.

MULTIPLE ROBOTS AND MULTIPLE AGENTS

John W. Perram

The Maersk Mc-Kinney Moller Institute for Production Technology
SDU-Odense University
Campusvej 55
DK-5230 Odense M
Denmark
Email: jperram@mip.sdu.dk

Abstract

There are two approaches to modelling the mechanics of robots, the (standard) Newton-Euler approach based on the use of generalised coordinates and the constraint dynamics approach which uses redundant coordinates and computes the forces of constraint explicitly.
Based on our experience with particle simulations, we have used the constraint dynamics approach, both in the Euler-Lagrange and Hamiltonian formulation because we believe that it has the following advantages:

- *A better ability to deal with robot complexity, since the method has been used to simulate molecules with many thousands of constraints for which it would be impossible to find generalised coordinates,*
- *Better performance, in that the matrix in the linear equations for computing the constraint forces is usually sparse with algebraically simple elements, whereas that for the matrix in the linear equations for the computation of the generalised accelerations is usually full with complicated elements involving expensive trigonometric functions.*
- *It leads to better software because the variables and equations are closer to the original physics so that OO and agent-oriented paradigms can be employed.*

This form of analytical mechanics has been combined with artificial force fields to automatically programme redundant robots involved in welding complex ship sections. The force fields are designed to ensure collision avoidance and task performance in a virtual world whose geometry is specified in the CAD model of the ship design. Deviations of the real world from the model one are detected by simple sensor systems and the programs corrected on the fly.

The notion of multi-agency enters in two ways. The form of analytical mechanics used views the robot as a collection of rigid bodies connected by joints which are modelled as constraints. The robot components can be thought of as a collection of agents which communicate with each other through the mutual forces of constraint arising from the external force fields. In this way, purposeful behaviour of the overall robot can be thought of as emerging from agent communication and the influence of the agent environment.

Introduction

Statistical Mechanics

Statistical mechanics is the branch of mathematical physics which deals with predicting or approximating the macroscopic behaviour of infinite-dimensional systems of Newtonian particles. The dynamics of such a system is described in terms of the Hamiltonian function

$$H(p,q) = T(p,q) + U(p,q)$$

where T and U are the kinetic and potential contribution to the total energy. An equivalent approach is the d'Alembert or Euler-Lagrange formulation. The evolution is described by the Hamiltonian differential equations
Interacting Hamiltonian systems with more than a few degrees of freedom are usually chaotic. However, under suitable conditions, a Hamiltonian system may converge to a region of

the (p,q) phase space which is "ordered" in some way, for example, a solid.

It is notoriously difficult to achieve analytic results for such complex systems, so that the main tools of investigation are molecular dynamics, which solves the equations of motion numerically and computes the macroscopic properties of the system as time averages, and the Monte Carlo method, which generates new configurations of the system by moving particles at random and computes macroscopic properties as ensemble averages over configuration space.

There are a couple of significant differences between the two approaches. Molecular dynamics is normally carried out at constant energy, so that properties such as temperature emerge as the time averaged kinetic energy. The Monte Carlo method proceeds at constant temperature, and in fact, suppresses dynamic information.

According to the ergodic hypothesis, properties for the same system computed by both methods should be the same. Interestingly enough, the Monte Carlo method has directly inspired the technique of simulated annealing, which has been used as an optimal search technique in artificial intelligence.

The first work suggesting that particle dynamics would be useful in artificial intelligence was due to Khatib [1], who achieved promising results with planning collision avoiding trajectories for simple 2-dimensional simulated robots by solving their equations of motion in an artificial force field attracting the tool center to a goal and repelling the robot from obstacles.

This work inspired us to try to use the method for automatically planning trajectories for real, redundant robots in 3-dimensions performing welding of geometrically complex components of very large ships being constructed at the Odense Steel Shipyard. To do this, we had to generalise Khatib's work in ways:

- Since the robot used was actually a composite 10 degree of freedom manipulator consisting of a standard 6-axis industrial robot and a 3-axis (xyz) gantry, we needed a more compact mathematical description than that contained in the usual description of mechanics in terms of generalized coordinates.
- We needed to design force fields to ensure that the parts of the robot and obstacles in the environment could be prevented from colliding. Both the parts of the robot and objects in the environment were represented in a standard graphics manner as surfaces consisting of polygonal patches.
- We needed to restrict the motion of the tool in relation to the task curve in the situation where the robot was actually welding.

The first and third problems were solved using the techniques of constraint dynamics which had been developed in the field of molecular dynamics. The second problem was solved by trial and error.

$$\dot{q}(t) = \frac{\partial H}{\partial p}, \dot{p}(t) = -\frac{\partial H}{\partial q}$$

Constrained dynamics

All but the simplest mechanical systems are constrained. Thus, the 2-dimensional simple pendulum, in which a point mass moves on a circle, has only one degree of freedom, because one has been removed by requiring that the distance of the mass from the point of support must remain constant. The dynamical effect of this constraint on the coordinates is a force along the string which ensures that the mass moves in a circle.

The information contained in this force is lost when the normal physicist's method of describing the system in terms of a single generalised coordinate, usually the angle between the string and the vertical direction, is employed. Constraint dynamics is a technique of formulating the problem using the original (redundant) Cartesian coordinates (x,y) of the point mass and explicitly solving for the constraint force.

Other types of constraint relevant to robotics are the restriction of the mutual motions of two component rigid bodies connected by revolute, prismatic and revolute joints imposed by those joints and the requirement that the tool center moves on the task curve. These constraints can all be represented as (usually quadratic) algebraic relations between the coordinates of the form

f(q)=a constant

or, in their time-differentiated form

Grad(f(q)).q'(t)=0

This equation has the geometrical interpretation that the velocity has no component in the direction of the normal to the surface in configuration space defined by the constraint, which means that the force associated with that constraint only has a component in the direction of the normal and the mechanical interpretation that the constraint forces can do no work. In fact, other, so-called non-holonomic constraints, such as those expressed by requiring that the instantaneous point of contact of a wheel rolling on a surface is at rest with respect to that surface have a similar Pfaffian form which we may write in matrix form as,

$J(q)q'(t) = 0$

but are not normally a total derivative, and therefore not integrable. We do not consider such constraints here as they do not arise in the applications described in this paper.

When there are many geometrical constraints, their intersection is a manifold of lower dimension equal to the number of degrees of freedom of the system and whose normal is a linear combination of their gradients, as are therefore the constraint forces. Efficient numerical methods exist for computing the constraint force components in the d'Alembert equations of motion

$Aq''(t) = \textbf{external forces} + \text{Transpose}(J)\ \mathbf{T}$

where **A** is the (usually diagonal) inertia matrix, by solving a sparse set of linear equations for the constraint forces **T**.

Force fields

The most well-known force fields, the Coulomb forces due to electric charges and harmonic forces which model springs, should actually be used with caution in this context. Coulomb forces are very long-ranged and can cause artefacts in finite sample simulations. Harmonic forces lead to linear equations of motion, and, if used to model rigid bodies, require a very short time step in the simulation.

These forces also have the property of being spherically symmetric, which makes them unsuitable for modelling artificial forces between the very non-spherical parts of robots and their surroundings. There are two strategies for designing efficient non-spherical forces. One is encapsulation, where we try to efficiently cover the relevant surface with a small number of encapsulating ones, such as ellipsoids [2]. The other is to compute the shortest distance between polygonal facets of the surface [3]. This is a quadratic programming problem for which efficient algorithms exist.

As the artificial forces act in a virtual world, we are not limited to simple examples from physics. As an example, Perram and Demazeau [4] modelled the force (actually the motive power or braking force) to be applied by a driver to a car so as to avoid collision with another car it is following as complicated functions of the relative position and velocity which imitated the way a human driver would behave. Thus it is possible to build in a certain amount of individual intelligence into these forces. It of course needs to be checked that the forces thus designed produce the desired sort of global behaviour.

Another example of a non-physical force is the so-called thermostatic force, which couples the kinetic energy of the system to a heat bath [5] whose kinetic energy or temperature can be controlled. This method can be used to move the robot's tool center smoothly onto the task curve or to control its velocity according to the demands of the process model as it moves along the task curve.

Mechanical agents

At this point, the author may be curious as to what all this has to do with agents. Consider the parts of a robot modelled as rigid bodies. At any instant, the state of each part is described by its coordinates, velocities and a set of parameters expressing its goals and beliefs in terms of force fields. At each step, these are updated according to the forces, or external influences, each part experiences. While the artificial force fields usually have the form of pair-wise additive contributions, the constraint forces and the thermostatic forces do not, since they are global properties of the system. Although these are conveniently and efficiently computed using numerical algorithms, they have the same character of a negotiation between the components. This line of thought indicates that there is a close connection between multi-agent systems and statistical mechanics [4].

Emergent trajectories

Assuming that the force fields have been suitably designed, numerical solution of the mechanical equations yields joint configurations at the rate of about 100 Hz. Although this is sufficient to plan trajectories in real time, the initial ship welding application produced an offline program valid in the virtual world defined by the CAD model of the work piece. Because of the high tolerances in shipbuilding, the small geometrical deviations of the real world from the CAD model could be measured on line using rather simple touch sensors and the program corrected.

Current versions of the system involve much more sophisticated sensors such as laser range finders or vision, so that perceived changes in the environment can be fed directly back to the CAD model so that the numerical simulation proceeds on line.

Bibliography

[1] O. Khatib, *Real-time obstacle avoidance for manipulators and mobile robots,* Int. J. Robotics Res. 5, 90-98, (1986)

[2] J.W. Perram, J. Rasmussen, E. Præstgaard and J. Lebowitz, *The ellipsoid contact potential: theory and relation to overlap potentials,* Phys. Rev. **E54**, 6565-6572, (1996)

[3] L. Overgaard, H.G. Petersen and J.W. Perram, *A general algorithm for the control of multi-link robots,* Int. J. Robotics Res. **14**, 281-294, (1995)

[4] J.W. Perram and Y. Demazeau, *A multi-agent architecture for distributed constrained optimization and control,* in Proc. SCAI'97, IOS Frontiers in AI, **40**, 162-175, (1997)

[5] W.G. Hoover, *Molecular Dynamics,* Springer Verlag, (1986)

COMPLEXITY PROBLEMS IN ANALYSIS OF MULTI-AGENT SYSTEMS

A. Slissenko

Dept. of Informatics, University Paris 12, France, e-mail: slissenko@univ-paris12.fr

Abstract

The goal of this talk is to try to identify complexity problems specific to multi-agent systems. Clearly, the multi-agent systems inherit the relevant traditional problems related to communicational complexity, security, learnability etc. But they have also their own problems implied by the individual simplicity of agents, their intensive interaction with the environment, their cooperativeness and adaptiveness. We assume that the computational resources of any particular agent are very limited, though the basic actions and tools to construct and modify the current behavior can be chosen from a sufficiently powerful set. The problem is to distribute these resources between agents and supply them with initial procedures of behavior and that of modifying their behavior which would permit the cooperative agents to accomplish repetitive tasks. A general principle of adaptation of the collectivity of agents to the environment, given a type of tasks to accomplish, is the principle of minimizing this or that criterion with the flavor of epsilon-entropy of metric spaces. Such a notion gives some basis to compare the quality of algorithms of behavior and the quality of algorithms of adaptation. Though the minimum of the chosen entropy-like criterion could be hard to reach, the diminishing of this criterion can be treated as an indicator of a good algorithm of adaptation. Within the same framework of basic notions one can give quantitative criteria for other features of agents behavior.

DATA AND KNOWLEDGE MINING IN MULTI-AGENT SYSTEMS

Nikolay G. Zagoruiko

Institute of Mathematics SD RAS, pr. Koptiug, 4, Novosibirsk, 630090, Russia. E-mail: zag@math.nsc.ru

Abstract

The modern methods the Data Mining (DM) and Knowledge Mining (KM) and possibility the using it for improving some features of agents and multi-agent systems as a whole are considered. At a microlevel the DM and KM methods can be used to supply to the agents skill and desire to reason by analogy, to conduct the self-analysis and to aspire to self-perfection. On macrolevel these methods can allow the agent to use intuition during the analysis the regularities of social behavior the other agents, rating their competence and objectivity at the collective solution of tasks.

Keywords: *data and knowledge mining, taxonomy, pattern recognition, filling gaps, prognosis, self-analysis, self-perfection and intuition.*

1. Introduction

Let's explain in the beginning what we understand under Data, Knowledge and Data and Knowledge Mining. Initially the data reflects the separate facts: "the house №1: a material - brick, height - 25 м "; "the house №2: a material - tree, height - 7 м, color of walls - gray", etc. After the analysis of these data it is possible to discover regularities of such kind: "The houses described in a database, are divided on height into three classes: < 5 m, 6 - 12 m and > 12 м "; " If (material - tree), then (height < 10 м) ". Our understanding of investigated objects became more general, we, as though, "have taken and have lifted on a surface" the regularities hidden in a set of the separate facts. The discovered regularities are formulated as easily perceived by the man and convenient for the further machine processing, and are referred here as "knowledge". The methods of the information analysis, providing the transition from the data to knowledge we refer as Methods of Data Mining (MDM).

We enter the metrics into knowledge space allowing to measure distances between any two knowledge [1]. It gives possibility to analyze knowledge contained in the Knowledge Base by the methods, similar to those used for data analysis. As a result, the regularities hidden in the Knowledge Base, i.e. Metaknowledge, of such type are found out: " the knowledge described in the Knowledge Base, usually contains no more than 7 predicates "; "concerning structure and values of predicates the knowledge are precisely divided into two classes: knowledge about village houses and knowledge about urban houses ", etc. The methods of discovering regularities on set of knowledge we call here as Methods of Knowledge Mining (MKM). The MDM are widely applied in various systems of artificial intelligence. Recently the MKM have begun to be applied as well. What can give their application for creation of agents with highly developed intellectual and emotional characteristics?

2. The agent of a partner type.

At a level of the separate agent the urgent task is to transform it from the passive assistant into the active partner of the user. What qualities should the good partner have? Apparently, each of us would like to have the partner, which would **reasonable**, i.e. have **wisdom** (to be able to choose the purposes, to put tasks), **mind** (to be able to build the effective plans of achievement of the chosen purposes) and **will** (to have desire and forces for achievement of the purpose). It will be well coordinated to the basic postulate of Russian philosophical school, that in all reasonable there is a base triad: an **idea, word** and **act.** Therefore, the agent of a partner type except the skill to fulfil the tasks of the user should have the following intellectual and emotional functions: under the initiative of formulation the reasonable tasks, to choose methods of their solution and to solve the formulated tasks.

The modern methods of DM and KM seem to be useful in achievement of such purpose. In particular, with their help the agent can constantly supervise a status of the information base, finding out various imperfections in it. At detection the gross errors or blanks in the data and knowledge it can stress attention of the user on them and offer the variants for correction of this mistakes and filling of blanks. Comparing available and new data and knowledge, the agent can find out the contradictions between data, knowledge or between knowledge and data and formulate an inquiry to the user or to other agents and experts to eliminate these contradictions. Finding out arising changes in regularities, the agent may focus attention of the user on them and offer the forecasts of possible consequences of these changes. At detection noninformative, duplicating, out-of-date data and knowledge the agent with the consent of the user can remove them in archive, thus raising reactivity. Making classifications of the data and knowledge, the agent may structure the information thus accelerating the process of a logic inference. What methods can help to realize such set of skills and wishes? Let's give the brief description of these methods.

3. Algorithms for improvement of the data table.

For filling gaps, discovering and correction the mistakes in the data table agent can use the algorithms of the ZET family [1]. Let us consider for simplicity, that initial data are of the type "object-property' table, where strings *(1,2,...i,...l,...m)* correspond to *m* objects *a(i)*, and columns *(1,2,...j,...k,...n)* - to *n* features or properties *x(j)*. Let us suppose, that value *b(ij)* of *j*-th property of *i*-th object is unknown for us and we want to obtain its predicted value. Let us name the elements, placed in *i*-th string and *j*-th column, intersection of which corresponds to predicted element, the "basic" elements. After normalization the values of each table column the competent matrix is chosen, including *r* strings, the most similar to the base string *i*, and *v* columns, most tightly connected with the base column *j*. As the measure of string similarity we may use distance between them in Euclid space of properties, and as the measure of intercorrelation (dependency) of properties - modulus of correlation coefficient. Each *l*-th string of the competent matrix obtains the weight *L(il)*, proportional to its competence, which increases with the decrease of the distance between *i*-th and *j*-th strings. In the same way, competence *L(jk)* of each *k*-th column increases with increase of its correlation with *j*-th column. *L* value varies from 0 to 1. The next step is construction of *r* linear regressions between *i*-th string and all *l*-th competent strings. From each of such regressions, knowing the value of element *b(lj)* in *l*-th string, one can obtain the variant of prediction of the missed element value in *i*-th string - *b(ij,l)*. Averaging of obtained variants with competence weights gives the variant *b(ij1)*, worked out with participation of all *r* strings:

$$b(ij1) = \sum_{l-1}^{r} b(ij,l) * L(il) / \sum_{l=1}^{r} L(il)$$

In the same way we determine the variant *b(ij2)*, obtained by averaging of predictions *b(ij,k)* from all *v* columns with weight, equal to

$$b(ij2) = \sum_{k=1}^{v} b(ij,k) * L(jk) / \sum_{k=1}^{v} L(jk)$$

competence *L(jk)* of these columns:

.Final variant of predicted value is obtained, for example, by simple averaging of predictions from strings and columns: *b(ij)'* = *{b(ij1)+b(ij2)}/2*. The level of trust to the received result can be estimated by the **dispersion criterion** [2]: **the higher is dispersion of the forecasts of sets *b(ij, l)* and *b(ij, k)*, the greater mistake of the final forecast *b(ij)'* will be.** It can appear that the available information is not sufficient for the sure forecast of absent value. In this case the agent will repeat attempts of filling of available blanks at each receipt of the new data. The modification of this base algorithm can be used for the solution of many other agent tasks. So, with the help of ZET the agent can predict all known elements of the data table and compare these predictions to actual values. If the distinction between the fact and forecast will exceed some threshold, the agent will inform the user about it. It can appear that it is the unique fact, which is dropping out of regularities in the given table. In this case user confirms its validity. It frequently happens, that the gross blunder is found out. In this case user either replaces erroneous value by the correct one, or charges the agent to find correct value. For performance of this task the agent uses all internal resources: expands structure of competent submatrix, applies knowledge from the knowledge Base, calculates required value with the help of the model of the investigated phenomenon – if such model exists. If the satisfactory result is not received, the agent puts aside a multi-agency network and searches necessary data, knowledge or model in accessible information bases of other agents. At last, through the appropriate agent, it can address to the user being the expert in interesting area, the request to state the judgement about the given fact. If as a result of all these actions it is possible to find the value well agreed with the available data and knowledge, the agent informs user about performance of the task. Otherwise, the agent in process of receipt of the new data and knowledge will repeat constant searches of the correct solution, reminding the user about the trouble in a Database. At each moment of time it is possible to find the sum *S* of distinctions between the actual data and their ZET-predicted values. The value of *S* can help the agent to indicate the Database perfection and stimulus for self-perfection. If *S* exceeds some allowed threshold value, the agent includes all mechanisms of correction of a situation described above in a time free from performance of the user task. ZET-methods allow separating the actual data from out-of-date information. The agent predicts elements of each line and analyzes "age" of lines contained in the competent submatrix. If the regularities of process inducing the data vary in due course, then to every line of age *T* there will correspond the set of competent lines with age near *T*. In result, for the current moment of time it is possible to find the greatest age *T'* of lines contained in competent submatrix. The agent can consider lines with age, larger than *T'* as "irrelevant" and transfer them from a working field to archive under the consent of the user. It raises reactivity of the agent and simultaneously serves for the user as the information on occurrence of essential changes in behavior of investigated object or process.

4. Methods for structuring the Database

Using algorithms of the **cluster analysis** [1] agent can separate the set of objects (lines) on k subsets (taxons *s1, s2,...sj,...sk*) and reveal the typical representative ("precedent") for every taxon, i.e. object *aj* with minimum the sum of distances from it to all other objects of the given taxon *sj*. The information on number of taxons, their sizes, structure and characteristics of precedents represents the brief description of the Database structure regularities. The structured Database facilitates the agent to solve more frequent tasks. So, if now user will need to find the object in a Database most similar on the properties to the given object *ai*, the agent does not need to compare object *ai* with all other objects. By the most similar precedent *aj* it is possible to find the nearest taxon *sj* and then carry out comparison the object *ai* with objects of it taxon only. If some other agent of multi-agent system will search for the necessary data it will not need to look at once through the contents of all Database of the all agents. It is enough to study structure of precedents to decide, whether it is possible to find interesting information here. During work of the agent the new data will be added to its Database. To support the structural order it is possible to use **methods of pattern recognition** [1]. In the simplest case the new object will be added to that taxon, which precedent is most similar to this object. In a Database the objects usually have the superfluous description. Among their characteristics one can find noisily, not informative, duplicating each other ones. The reasonable agent-partner should eliminate these lacks. It is possible to select the most informative subset *n* of the characteristics from initial set *q*, for example, by means of **algorithm DTSA** (Directed Taxonomic Search of Attributes) [1]. The measure of distance ***R*** between the characteristics is reverse proportional to their interdependence. So, knowing the distance *R* between the characteristics it is easy to divide them on *n* of groups (taxons). Then the typical representatives (precedents) of every taxon are determined. The algorithms of taxonomy guarantee that the dependence between precedents will be minimal. A subset of the *n* most informative characteristics is produced in such a way.

5. Methods for self-perfection of the knowledge base.

Each element of knowledge of the type "$x=3\div5$" contains the statement that the probability of events $x<3$ and $x>5$ is equal 0, and the probabilities of events $x=3$, $x=4$ and $x=5$ are equal 1/3. So the task of the measurement of distances between knowledge is reduced to the task the measurement of distance between distributions of probabilities. Let us consider that there is a statement *ax* for the variable *x*. Let's divide the scale of possible values of *x* into *m* parts with density of probability in each part equal to *1/m*. Let's record a positions of borders between parts: *xa1, xa2,...xai,...xam*. Let's consider the second statement about variable *x* - *xb*. Having done the same procedures with them, we shall receive borders *xb1, xb2,...xbi,...xbm.* If these statements are identical, the appropriate borders will coincide with each other. At the different statements of border the values *xai* and *xbi* will settle down in different points of an axis *x*. The sum of such distinction *r* defines one of the parties of distinction between the statements. At same average distance between distributions, the statements can differ from each other by categoriality. Than more distribution *xa* differs from uniform, the higher is categoriality. If we will find entropy of uniform distribution and distribution *xa*, their difference *ha* can serve as a measure of a categoriality of judgement *xa*. The measure of a categoriality of the second statement *hb* is obtained in the same way. In result the average categoriality of the two statements is accepted equal to $h = (ha+hb)/2$. The special importance has the presence of sites of an axis x, concerning which judgement *xa* and *xb* are identical. These zones of the consensus can be used for the elimination of the contradictions in the knowledge Base. The higher are distinctions of two distributions, the less the zone of consensus *v* is. Let's accept, that the third component of distance between distributions is equal to $w=1-v$. In result the measure of distinction (distance) between two statements can be expressed by the value $R=r*h*w$.

Using this approach, we can find distance between knowledge and thus create the conditions for application of KM methods similar to DM methods described above. The taxonomy of knowledge makes the Knowledge base (KB) more transparent. The selection the precedents of the taxons accelerates procedure of a logic inference. At filling KB it is important for the agent to find out the contradictions in knowledge. With this purpose the agent defines distance between conditions (" If ... ") *Ri* and consequences (" than ... ") *Rt*. If *Ri* is small, and *Rt* is great, it means, that the strongly distinguished conclusions are done on the base the close conditions, and the agent informs the user about the noticed contradiction.

The metrics in knowledge space allows overcoming essential lack of the programming languages of a PROLOGUE type. Now it is possible to make conclusions not only on strictly identical, but also on "similar" knowledge, simulating a human way of thinking by analogy.

There are algorithms of pattern recognition, selection of a subset of informative predicates, detection of gross blunders and filling of gaps in KB [1], which allow the agent not only to support KB in a good form, but also to improve it during operation.

6. Agent-partner in multi-agent environment

The reasonable agent should be the good partner not only for the owner (user), but for other agents as well. It should be attentive to requests of other agents, inform them about important information events and carry out the honest share of work in the collective solution of tasks. The good reputation will help the agent in the solution of its own tasks with application the resources of other agents. At the same time, it should well be guided in multi-agent environment: to know specialization of the separate agents, their competence, structure of mutual relation between the agents, to be able to predict results the solution of the tasks with participation of other agents. The paper at our previous conference was devoted to the questions of an automatic rating of competence of the agent [3]. The analysis of the structural characteristics of multi-agent system is considered at the given conference in the paper [2]. The DM methods for detection of coalitions in agent environment, rating the competence of each coalition, classification of tasks solved by system, and forecasting the results of the solution of new tasks by separate agents coalitions and system as a whole are described there.

7. Conclusion

The methods of Data and Knowledge Mining give wide opportunities for creation of reasonable multi-agent systems consisting of the agents a partner type. The brief acquaintance to these methods can be received at Web site http://www.math.nsc.ru/AP/oteks.

Bibliography

1. Zagoruiko N.G. Applied Methods for Data and Knowledge Analysis. Ed. by Inst. of Math., Novosibirsk, 1999.
2. Zagoruiko N.G. and Zhuravlev Yu. I. Decision Making in Multi-Agent Systems. Proc. Of This Conference.
3. Zagoruiko N.G. Methods of Competence Estimation of the Agent in the Multi-Agent Network.

http://spase.iias.spb.su/ai/endlish/windex.htm

Work was supported by Grant of the RFFR № 99-01-00582.

PART II
Regular Papers

PHYSICS OF COLLECTIVE INTELLIGENCE: MASS FLOW AND SMART AGENTS FOR LOAD BALANCING IN COMMUNICATION NETWORKS

Andrew Adamatzky and Owen Holland

Intelligent Autonomous Systems Laboratory
University of the West of England
Du Pont Centre
Frenchay Campus
Bristol BS16 1QY
United Kingdom
Telephone +44 (0) 117 9656251 ext 2662
Email: Andrew.Adamatzky@uwe.ac.uk *and* Owen.Holland@uwe.ac.uk

Abstract

This paper is an attempt to characterise some aspects of the new wave of reaction-diffusion and ant based computation, and to discuss their place in the class of fully distributed load-balancing algorithms that solve the dynamic load balancing problem of communication networks. The main question of the paper states: what are the advantanges of the intellectualisation of the control agents and how much do we pay for the smartness? We start our investigation with random walk techniques and the electricity paradigm, carry on with the reaction-diffusion approach, and finish the construction of the computational hierarchy with the ant paradigm and smart agents.

Keywords: *collective intelligence, combinatorial optimization, distributed routing, communication networks, complexity, distributed intelligence, mobile agents*

1 Introduction

The idea of computing with a swarm of simple agents can be developed in the direction of using enormously large numbers of elementary entities which have minimal physics based attributes - the ability to diffuse, flow, or randomly walk through a space or over a graph or network, possibly being affected by some local conditions. In the limit this reduces to electricity or chemical diffusion. An example is field computing, where a graph is expressed as a network of resistances, and the flow of current in response to applied voltages is interpreted as a solution to some difficult problem. Such methods were well developed in the area of operational research from the 1950s onwards. In a recent example, Vergis *et al* [38] showed how, by representing spare node capacity in a communication network by conductances, it was possible to instantaneously calculate all shortest paths. Marshall and Tarassenko [30] used a similar technique for path planning problems in robotics, showing potentially enormous gains in speed over serial computation, and planned to implement the solution in VLSI. Chong (1993) [15] applied a similar solution to problems of optimal routing in packet switched networks. Scarcely more complex are the reaction-diffusion systems of real or simulated chemicals which diffuse away from nodes and form immobile precipitates when they encounter different chemicals from other nodes; these systems can construct various useful spatial structures such as Voronoi partitions and skeletons (e.g. [2]). This class of models was probably first developed by Tsetlin [37], Stefanyuk [34] (see historical review in [35], and Rabin (see [17]). In particular, Tsetlin's book [37] identifies many of the features associated with biologically inspired automata: randomness, self organisation, and distributedness. The absence of direct interactions between agents was also implied. These principles were applied to the same types of problems as seem attractive now - to the control of telecommunications networks [13] and groups of radio stations [34]. Rabin also introduced the ideas of jumping and walking pebble automata that could solve problems on graphs and lattices by interacting with the consequences of their previous actions using what is now called stigmergy.

The ant based computation paradigm was developed in mathematical biology and biology inspired robotics [10] independently of primary physical and automata models, and only recently has moved toward the mainstream of distributed computation models (see e.g. [16]). The basic principles underlying the possible use of ant algorithms for the control of telecommunication networks were set out in [9]: mobility of the agents, interaction mediated by the environment with no direct contact between the agents, decay of messages with time, stochastic components of agent behaviour, and large numbers of agents. These principles can be discerned in more recent works on network control [33, 32, 11] and distributed combinatorial optimization [16, 22, 26]. Another significant area of application of ant-like algorithms and computation with mobile agents deals with problems of computational geometry, particularly the approximation of proximity graphs (see e.g. [4, 5, 6]).

The paper explores the potential of some of the new methods for controlling packet-switched networks with dynamically changing topology and variable traffic patterns. The routing tables controlling packet switching are updated by mobile agents without any central control and with no *a priori* knowledge of the topology of the network.

2 Random walk, electricity and approximation of trees

The connection between random walks and electrical flows was established several years ago (see [23] for detailed overview). The principal results state that various classes of trees can be approximated by random walks, and that random walks are equivalent to electrical flows. Let $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$ be a finite graph with vertices from $\mathbf{V}$ and edges from $\mathbf{E}$. If we start a random walk at x_0 and absorb it at $\mathbf{Z} \subset \mathbf{V}$, and S_{xy} is the number of transitions from x to y then $\mathcal{E}[S_{xy} - S_{yx}] = I_{xy}$, where I is the current when a potential is applied between x_0 and $\mathbf{Z}$ [28]. Moreover, if $\mathbf{G}$ is a connected graph and V is a voltage function derived from the unit current flow from x_0 to ∞ then for any x there is a path of vertices from x_0 to x along which V is monotonic [28].

The probabilistic interpretation of electrical current is quite clear [28]: particles enter the graph at x_0; they do Brownian motion; they spend more time on edges with smaller conductances; they are removed when they hit any vertex from the set $\mathbf{Z}$ of destination vertices; and the net flow of particles along an edge is the current on that edge.

Let weights be equivalent to conductances (or resistances), and the batteries be connected between $\mathbf{S}$ and $\mathbf{Z}$, $\mathbf{S}, \mathbf{Z} \subset \mathbf{V}$, in such a manner that the voltages (potentials) are 0 at $\mathbf{S}$ and 1 at $\mathbf{Z}$. Current then flows along the edges. Ohm's law says that for any edge $(x, y) \in \mathbf{E}$ the following relation between current and resistances will hold:

$$I_{xy} = \frac{V_x - V_y}{R_{xy}},$$

where I_{xy} is the flow along (x, y) and R_{xy} is the resistance of the edge. By Kirchhoff's node law we have:

$$V_x = \frac{1}{deg(x)} \sum_{y:(x,y)\in\mathbf{E}} C_{xy} V_y.$$

That is, the voltage function represents the solution of the Dirichlet problem [28]. Moreover, Kirchhoff's cycle law also holds. If path $x_1, x_2, \cdots, x_n, x_1$ is a cycle then

$$\sum_{i=1}^{n} I_{x_i x_{i+1}} R_{x_i x_{i+1}} = 0.$$

In words we can say that if a voltage is established between vertices x_0 and x_1 such that it is 0 at x_0 and 1 at x_1 then the voltage at vertex x equals the probability that a random walk visits x_1 before it visits x_0 when it starts at x.

Let us look now at the well known algorithm for the generation of random trees as represented in [28]. The algorithm is similar to a probabilistic technique known almost a century ago – the Bienamé-Galton-Watson construction of random trees. Given a set of probabilities we begin with one individual node and let it reproduce itself (make connections with neighbouring nodes) according to the given probabilities; we then let each such child reproduce itself in the same way *etc.* Lyons and Peres [28] discuss how to choose one of the random trees from the numerous trees of the given graph by using some properties of Markov chains. In the case of a directed graph, the spanning tree is a subgraph that includes any vertex of $\mathbf{G}$ and has a root vertex x_0 such that every other vertex is the tail of exactly one edge in the tree. To choose the tree we pick up some vertex x_0, and another vertex y, and draw the path from y to x_0 (it exists by irreducibility); then pick up another vertex z not contained in path from y to x_0 and draw the path from z to x_0 and so on. We continue until all vertices have been used. The path can be chosen probabilistically according to the weights of the edges. If all weights are the same the approximated tree will be chosen with uniform probability. If the weights are different, and if at every step we choose the edge with the locally minimal weight, the minimum tree will be approximated.

The basic result noted by Aldous [8] and Broder [12] states that *a random walk on a finite connected graph constructs the uniform random spanning tree.* This is because a random walk on $\mathbf{G}$ is seen as the discrete time Markov chain with transition probability matrix P calculated as follows: $P_{xy} = deg(x)^{-1}$ if $(x, y) \in \mathbf{E}$, otherwise it is 0.

These ideas provide the basis of algorithms for the approximation of spanning trees during neuron morphogenesis described in [1]. The tree is approximated by the growing branches of the dendritic tree. The growth cones of different branches move at random and compete for occupation of given points. The *energy* of the growing cones of the branches decrease with time. The branch that has maximum energy wins.

3 Ant paradigm

The key features of ant based methods centre round minimum entities called ants which can move through some spatial domain which is typically a continuous or discrete Euclidean space, a graph of nodes embedded in a Euclidean space, a graph corresponding to some topological or topographical domain, or a graphical representation of a problem in some non-spatial domain; graphs may be directed or undirected. Each ant may carry with it a set of attributes and variables which may be affected by what it encounters as it moves over the graph: typical attributes are source, destination, and type; typical variables are age, distance travelled, and route taken. The ants move according to rules which may take into account any of the following: fixed local qualities of the domain; encounters with other ants; local qualities of the domain which may have been directly altered by ants; local qualities of the domain resulting from some autonomous process, which may be affected by alterations made to the domain by ants; and hard constraints which may be required for the particular target problem. The determination of movement is typically stochastic rather than deterministic, with the likelihood of choosing a particular direction being a function of the ants internal state, and local characteristics, which may include fixed factors, and factors due to or influenced by ant activity; it may employ in addition a fixed or variable amount of noise. An ant may be brought into being at a time and at a location determined by external factors (for example elapsed time), or by local factors, or by another ant subject to some contingency. Ants may terminate under certain conditions; for example, when they reach their designated destination node; termination may be accompanied by changes to the domain (such as updating segments of the ants route) or launching new ants which may inherit information from the terminating ant. An ant may alter the local qualities of the domain in ways, at times, and by amounts, which may include the following: the quantity modified may be a node or edge, or may be associated with a directed or undirected edge, and with the source, destination, route, or other characteristic of the ant, and may be at the present position, the immediately previous position, or at every previous position; the time of modification may be immediate, or after termination of the ant; the amount of modification may be fixed, or may vary with some local or global attribute, or may vary with the absolute value of some attribute or variable internal to the ant, or with the relative value in comparison to other ants. The effects of any such modification may persist indefinitely until changed, or may change autonomously with time. The solution to the problem may be the effect on some autonomous process of the activities of the ants, or may be a static configuration derived from the modified domain, perhaps by probing with a mobile agent making deterministic movement decisions. This degree of complexity must clearly support a number of distinct phenomena affecting the results, and since the list has been compiled from a relatively small number of algorithms, each algorithm probably depends on the operation of more than one phenomenon. Ant based methods can be used as alternatives to established methods for simple problems, or for intractable problems where conventional algorithms are unsatisfactory. The types of difficult problems for which ant algorithms seem to be particularly well suited are those combinatorial optimisation problems, which can be expressed in a suitable graphical form. All have the underlying characteristic that ants associated with each part of the problem or solution both affect and are affected by one another, principally by using and modifying route segment information on the graph.

Behaviourally, ants are usually considered to be rather simpler and less competent individually than solitary insects. They are dominated by specific sensory inputs, and often behave as if implementing simple if-then rules and functions, but with a strong admixture of random behaviour. Although short and long term learning are often used, and some real ants appear to become attached to particular tasks, the application of an agent-based model using beliefs/desires/intentions is wholly inappropriate; ants are essentially automata. However, any implementation of interacting high level agents which uses large numbers of space or graph-based mobile agents which can interact directly or indirectly via environmental modification will inevitably show some lower level mass behaviours; a nice recent example is [24] which deals with pedestrian crowds high level agents behaving like simple automata.

4 Diffusion-like algorithms

The *echo* algorithm by Ahuja and Zhu [7] was one of the first diffusion based techniques for the construction of the minimum weight spanning tree. During the execution of the algorithm messages are sent by the *root* down the tree (down wave). When the messages arrive at the leaves they are turned into echoes and run back up the tree (up wave) [7]. In the down wave the root sends information to all the nodes, and it receives information collected from all the nodes in the up wave phase. The down and up wave are repeated until the minimum spanning tree is constructed. We discuss now how a similar technique can be realised in reaction-diffusion systems.

Let $\mathbf{G}$ be a lattice where every node is connected by undirected edges with its four closest neighbours. The state of every node x at time step t is characterised by the tuple $< c_x^t, p_x^t >$ where c_x^t is the concentration of a reagent and p_x^t is the orientation of a pointer. The concentration variable takes their values from the real interval $[0,1]$ and p_x^t points either to the neighbour y, $p_x^t = y$, $(x,y) \in \mathbf{E}$, that is the closest to the root x_s of the spanning tree that is under construction, or nowhere, $p_x^t = \cdot$. Initially all nodes except x_s have no reagent and their pointers are not oriented, i.e. for any $x \in \mathbf{V}$ we have $c_x^{t=0} = 0$ and $p_x^{t=0} = \cdot$, whereas $c_{x_s}^{t=0} = 1.0$. In our rather clumsy notation the diffusion equations can be written as

$$\dot{c_x} = \delta\Big(\sum_{y:(x,y)\in\mathbf{E}} c_y - c_x \Big),$$

where $\delta = 0.25$. Space is therefore discrete, but time is continuous.

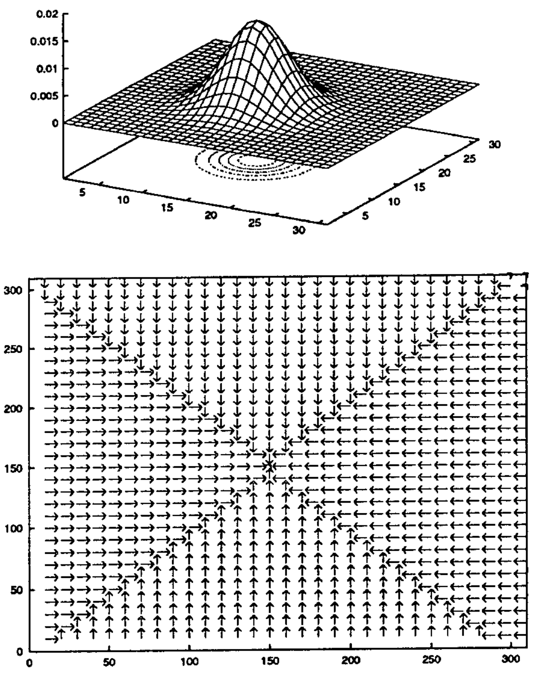


Figure 1: Concentration profile (top) and orientations of the pointers (bottom) at the 30th step of the reaction-diffusion tree constructor.

The pointers change their states by the rule:

$$p_x^{t+1} = \begin{cases} y: & (x,y) \in \mathbf{E} \quad \text{and} \quad c_y^t = \max_{z:(x,z)\in\mathbf{E}}\{c_z^t\} \text{ if } p_x^t = \cdot \\ p_x^t & \text{otherwise} \end{cases}$$

An example of the concentration profile and pointer field for the case of static uniform weights of edges is shown in Figure 1; for $x_s^{t=0} = 1.0$, $\mathbf{G}$ is a grid of 31×31 nodes, $x_s = (15,15)$, and $\delta t = 0.1$; the results are shown at the 30th step of the simulation.

In this version the distance minimal tree is approximated. However, we can also take loading (or spare capacity as the inverse of loading) at the nodes into consideration. Let σ_x be a spare capacity at node x. Then the diffusion equation takes the form

$$\dot{c}_x = \delta\Big(\sum_{y:(x,y)\in\mathbf{E}} c_y - c_x \Big) - \sigma_x^{-1},$$

assuming $\sigma_x > 0$ and $c_x := 0$ if $c_x <= 0$. After such modification the maximum spare capacity of the spanning tree rooted at x_s is approximated. An example of a numerical simulation

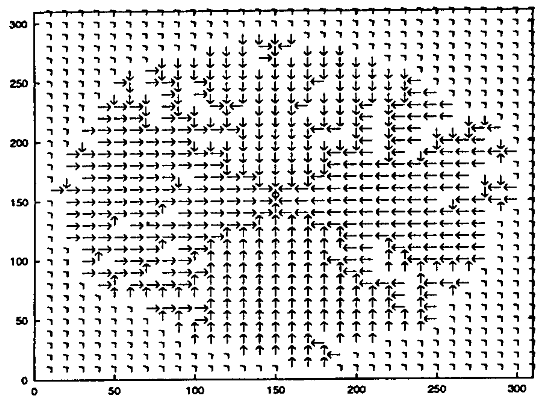


Figure 2:

is given in Figure 2 ($c_s^t = 500.0$ every time step and σ_x is a random variable distributed uniformly in the interval $[1, 10.0]$).

As we see from the pointer equation the state of the pointer is changed only once. That is, the technique is not adaptive. To make it adaptive we have to reset the pointer states before every round of the diffusive approximation of minimum weight/capacity trees or spread the orientation of a pointer amongst all neighbours of the node, making it fuzzy or probabilistic.

We can now discuss routing tables rather than simply deterministic pointers. The routing table based at node x of grid $\mathbf{G}$ has four entries (one entry per neighbour node because we are approximating a tree with a *given root*) per root x_s. Here we still consider a single root x_s (that is we are interesting in only sending packets to node x_s) therefore we deal with the real valued vector of order 4.

Let us use now three reagents, α, β and γ. The α reagent diffuses from the root node in the same manner as it did in the former model. Imagine now that every edge (x, y) consists of two tubes (x, y) and (y, x) filled with reagent β. β is a catalyst:

$$\alpha + \beta \rightarrow \gamma + \beta,$$

the γ reagent is produced as the result of the reaction between α and β. The concentration e_{xy} of γ at tube (y, x) of edge (x, y) reflects the amount of α reagent that has passed through

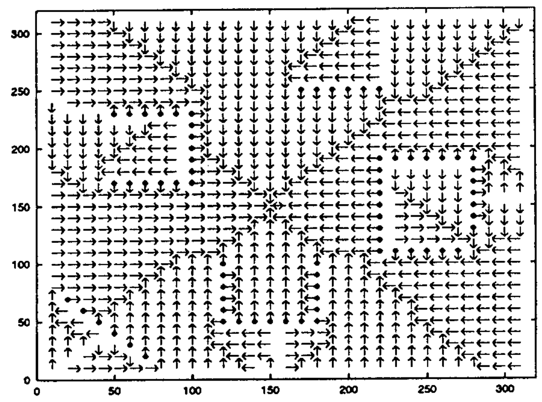


Figure 3: Overloaded nodes (diamonds) and orientation of pointer in the reaction-diffusion model of load balancing.

the tube:

$$\dot{e}_{xy} = f(c_y - c_x)$$

The numerical implementation of function f can vary from model to model. If we need to know which node a given node x is connected with at time t of the system's evolution we, can compute the state of the pointer as

$$p_x = y : e_{xy} = \max_{z:(x,z)\in \mathbf{E}} e_{xz}.$$

If we are anxious about the growing values of e_{xy} we can make them "probabilistic" via normalization:

$$e_{xy} := \frac{e_{xy}}{\sum_z e_{xz}}.$$

Now we can establish the state of the pointer at every real step of the evolution time. Let us check again how good this technique is.

Using spatial analogies we can think of overloaded nodes as obstacles placed on the grid $\mathbf{G}$. Several different versions of obstacles are represented on a grid (Figure 3, overloaded nodes are diamonds) by groups of nodes with $\sigma_x = 2$; the rest of the nodes have $\sigma_x = 100$.

A drop of reagent is placed at node $x_s = (15, 15)$: $c_{x_s}^0 = 100$. After a certain amount of diffusion time the distribution of the pointer orientations (Figure 3) finds the optimal tree rooted at x_s.

Moreover looking at the dynamics of the diffusion we see that the reagent flux through the

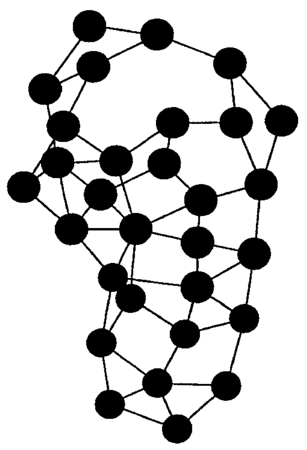


Figure 4: Topology of SDH network of British Telecom used in the simulations.

obstacle-nodes is virtually blocked and almost all the diffusion goes through the "open" parts of the grid.

So, we know that a spanning tree is approximated using diffusion, that this is the minimum weight tree when the nodes or edges are weighted, and that it can be dynamically recomputed at any time of the systems evolution. Time is continuous, therefore the system reacts to changes in loading or spare node capacity.

Let us introduce a second entity that moves along the branches of the tree towards the root. This may be the diffusion of another reagent, or the movement of packets. We know from the introductory sections that one can hardly see the difference between diffusion and random walk, or diffusion along gradients and packet flows; however, as soon we as deal with communication networks, we have to consider packets of information that are discrete entities.

5 SHBR-algorithm: ants on the net

Here we used a modified version of the Schoonderwoerd - Holland - Bruten - Rothkrantz (SHBR) algorithm, initially designed for circuit switched networks [32, 33], on a packet switched network.

The original version of the SHBR-algorithm was tested on the British Telecom synchronous digital hierarchy network (Figure 4). Fro comparison with previous results we also run our algorithms on this network.

In this section we aimed to find the answers to the following questions:

1. does "discrete" reaction-diffusion do well when the numbers of diffusing components are limited?
2. what are the advantages of auto-catalytic self localisation of minimum load routes, using the pheromone paradigm?
3. do probabilistic agents do better than deterministic ones?

The basic model considered here, the $S1$-model, involves two type of messages: data packets, emitted with some defined generation rate, and control packets, or ants, which are generated at the rate of one ant per node per time step. Every ant can be represented by the tuple (x_s, x_d, x, a), where x_s and x_d are the source node and destination node of the ant, x is the current position of the ant in the network, and a is the so-called *age* of the ant. Every node x of the network has a routing table R with entries that look like R_{xyx_d}. The entries of the routing table are updated by the ants:

$$R_{xyx_s} \leftarrow R_{xyx_s} + f(a^{-1}),$$

so that some quantity proportional to the age of an ant is added to the appropriate entry. The updating of routing tables by ants is an analog of pheromone laying on ant trails.

At every step of the simulation the routing tables are normalized to make them look like probabilities: for every x_d

$$\sum_{y:(x,y)\in \mathbf{E}} R_{xyx_d} = 1.$$

Ants may also age when they encounter queues at nodes: $a \leftarrow a + g(l)$, where l is the length of a queue. Traveling to destination x_d, and being at node x at the current step of simulation time a packet is routed to the node y neighbouring to x where the entry R_{xyx_d} is maximal amongst all other entries R_{xzx_d}, $(x, z) \in \mathbf{E}$.

To test our ideas we designed three additional versions of SHBR-algorithm:

- *S2-model.* The ants move as in the $S1$-model but they do not have load dependent age. Age is simply incremented by distance measured in nodes: when an ant leaves its current node x its age is incremented, $a_x \leftarrow a_x + 1$.

- *S3-model.* The ants perform as in the $S1$-model except that they choose the next node at random. Being in node x at time step t, an ant goes to node y with probability k^{-1}, where k is the degree of x.

- *S4-model.* The ants follow their routing tables deterministically. That is, the ant (x, x_s, x_d) chooses to go to node y corresponding to the maximal entry of the routing table, i.e. y is such that

$$R_{xyx_d} = max_{z:(x,z)\in\mathbf{E}} R_{xzx_d}.$$

  In all except this, the ants behave in the manner as in $S1$-model.

Here we present some results on the simulation of ant based control on the BT topology network (Figure 4). The parameters of the simulation were chosen to conform in general the call and ant generation parameters in the Schoonderwoerd *et al* model [32, 33]. In our toy model

1. every node of the network has a capacity of 40 packets equally split amongst the queues of the outgoing links;

2. the capacities of the incoming links are not taken into consideration;

3. the packets are launched from randomly uniformly chosen nodes toward destination also chosen at random;

4. at every step of simulation time no more than 1.2K packets can be on-line (including packets waiting in queues and newly launched packets);

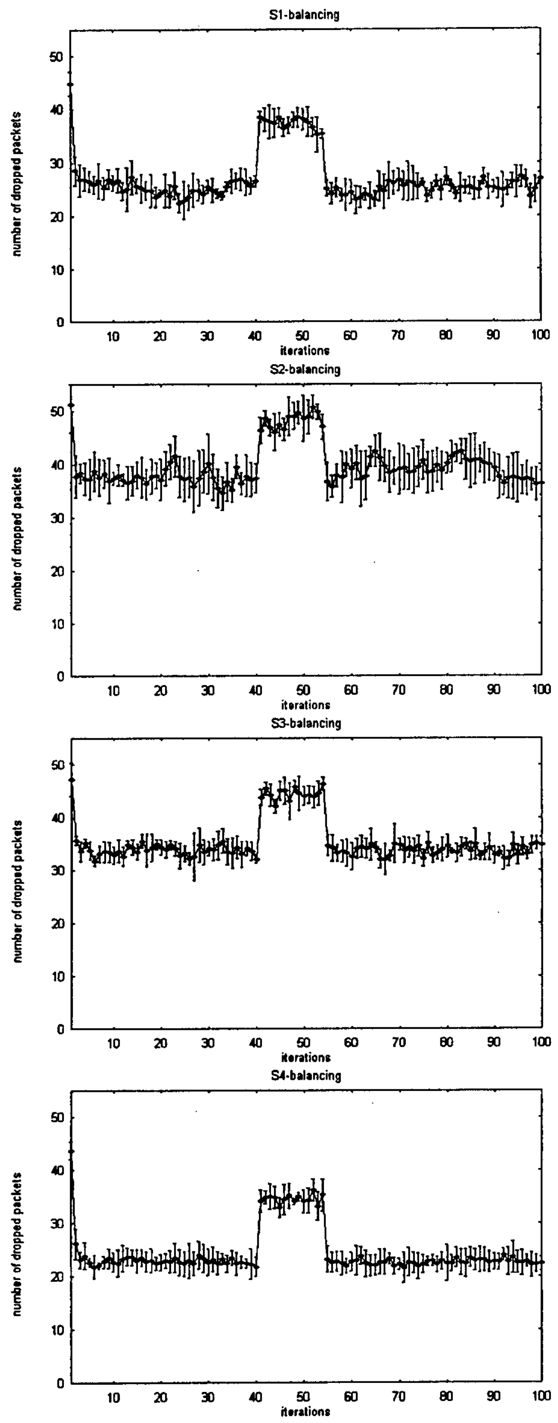


Figure 5: Mean number of dropped packets in $S1$-, $S2$-, $S3$- and $S4$-versions ant algorithm. Vertical lines represent standard deviations.

5. every node generates one ant per time step and ant is assigned destination node chosen at random,

6. we now also able to change the call patterns for a *hot session.*

Simulation is run for 10K units of time, and results are averaged for every 100th step of the simulation. In the [4000, 4500] time interval a hot session is set up on the network. In the simulation we present here, the hot sessions were organized as follows. Every time step node y is chosen to be the destination for every packet generated at this time step with a probability 0.3; any other nodes of $\mathbf{V} - \{y\}$ may be the destinations of packets with probability 0.7. The number of packets dropped at each time step is chosen to be the measure of efficiency of the load balancing scheme.

As we see from Figure 5 the effectiveness of these four techniques varies significantly.

Using the mean number of dropped packets as a parameter we can order the models in the following hierarchy: $S4$ (24.4 packs/step), $S1$ (26.8 packs/step), $S3$ (35.3 packs/step) and $S1$ (39.8 packs/step).

The huge gaps between $S4$ and $S1$ on one side and $S2$ and $S3$ on the another prove that simple mass transfer does not work when the number of control agents is small.

That is, there appears to be a trade-off between number of agents and the information passed by a single agent. The agents which are aging proportionally to the loading of the queues outperform the agents whose ages are simply incremented by a factor of 1.6.

When we consider the so-called *shock response* (the increase in number of dropped packets during hot session compared to that during normal operation of the network) we find that the mass flow modifications ($S2$ and $S3$ models) are quite tolerant: 8.2 packs/step and 9.2 packs/step. The deterministic ants are slightly better than the probabilistic ants (10.1 packs/step and 10.8 packs/step, respectively). Surprisingly, the networks controlled by the probabilistic ants ($S1$-model) exhibit more high amplitude (and certainly aperiodic) oscillations than the systems managed by the deterministic ants ($S4$-model).

Finally, what about the adaptation to the changing environment? If we compare the behaviour of the systems during the hot session we notice that the systems with probabilistic ants start adapting (i.e. number of dropped packets gradually decreases) immediately after the start of the hot session (Figure 5, $S1$). This does not happen with deterministic ants (Figure 5, $S4$), which appear to have little or no adaptive abilities.

6 Discussion

Despite numerous results and expressive demonstrations we are still quite far from a solid theory describing the connections between mass flow techniques and computations with mobile agents. In this last section we identify several points that appear to be important for further consideration.

6.1 Diffusion

As we saw in previous sections the discrete diffusion algorithms are similar to the technique used in the experiments done by Subramanian *et al* [27]. In their experiments all ants are uniform; they move at random on the network, and they die when their fixed life time expires.

6.2 Balancing of multi-processor networks

Another form of load balancing occurs in multi-processor networks, and involves finding a way to move tasks among processors to make the numbers of the tasks approximately uniformly distributed amongst the processors. The tasks should usually walk (i.e. move between the adjacent nodes) but not jump. The *diffusion schedule* states that the tasks have to be moved along edges depending on the number-of-tasks-gradient at the edge [18, 20, 21, 31]. The method is claimed to be asynchronous affine, fault tolerant and topology stable [20]. The simplest scheme looks like a typical model of diffusion on graph. Writing

ψ^t for the transposed vector of the whole workload on the nodes of **G** at time step t we have $\psi^{t+1} = \Delta\psi^t$ [1], where Δ is the doubly stochastic and symmetric matrix with entries from $[0, 1]$. The number of tasks moved between nodes i and j along the edge $(i, j) \in \mathbf{V}$ equals $\delta_{ij}(\psi_i^t - \psi_j^t)$. The scheme converges if the second largest eigenvalue of the diffusive matrix Δ is less than 1 in absolute value. We can conjecture that the rate of convergence is inversely proportional to the diameter of the graph **G**; thus, e.g. it is inversely proportional to the number of nodes in the case of a linear graph [40, 41] and should obviously be two times less when a linear graph or a mesh is formed into torus.

At the same time in the ant-based model, an ant at one node walks to an arbitrary uniformly chosen neighbouring node, and never stays at the same node for the next time step (except in a queue, i.e. at the exit). How does this relate to the diffusive schemes? In [20] it was demonstrated in numerical experiments that for different families of graphs with standard topologies the entries of the diffusion matrix are the same, and that they are inversely proportional to the degrees of the nodes. There are no general principles for the determination of the number of tasks left on a node at every iteration, so, the *leave no ants* principle seems to be a good assumption. It was also proved [40], that the convergence-rate-optimal diffusion coefficients have a form $\frac{1}{2d}$ when the network is a d-dimensional cube.

6.3 Reinforcement learning

The so-called distributed reinforcement learning scheme for load balancing [29] utilizes an *estimated delivery time* which the nodes exchange locally. A node sends a message to its neighbour with the estimated lowest delivery time, and asks its neighbour to send back an updated estimate. Its neighbour does the same when sending the message to its neighbour, and so on. When at a distance p nodes from the target node x_t the node x_s will finish the phase of updating after p steps. We can easily reformulate it in terms of ants: ants traveling from x_t to x_s are messages that deliver information about the estimate and inform the data messages about it by leaving pheromone in the routing tables. In terms of reaction-diffusion: the estimated delivery time along a path is proportional to the gradient along this path and it is reflected in the concentration of the precipitate at the edges.

6.4 Complexity

Here we write m for $|\mathbf{E}|$ and n for $|\mathbf{V}|$. The communication and time complexity of the construction of spanning tree is quite obvious. We need $O(m)$ messages to traverse m edges, and it takes $O(n)$ time steps because **G** can has diameter n. One of the first algorithms for the distributed construction of spanning trees was published in [25]; in the execution of the algorithm the fragments of tree formed the forests of trees and the minimum outgoing edges between the fragments and combinations of fragments via minimum weight edges are carried out repeatedly. It requires $(2m + 5n \log n)$ messages and $(5n \log n)$ time steps. Gallager's *et al* [25] algorithm was modified in [14].

There the network starts its evolution when every node knows the weights of its adjacent edges and finishes the computation with the spanning tree. The nodes exchange messages of different sorts. The message complexity has upper bound $O(m + m \log n)$ and time complexity $O(n \log n + 3n)$; in the synchronous version every message contains at most one edge weight or one node identity and $(\log n + \log \log n + 9)$ bits. The message complexity in the echo algorithm [7] is $(2m - 2(n-1) \log(n/2))$; the time complexity is $(2d \log n)$, where d is a diameter of **G**.

The *excitation* algorithm [3] for the construction of the spanning tree needs $O(m)$ messages and $O(n)$ times steps in the worst case (actually $\sum_{t=1}^{T} S_t = n$, where T is the time complexity of the algorithm and S_t is a number of messages generated at the t-th time step). When averaging the time complexity

[1]we can write it in the more familiar form $\psi^{t+1} = \psi^t + \Delta\psi^t$, assuming $\delta_{ii} = 0$

we can use the boundary $O(\log n)$ [36] in a very wide range of probabilistic assumptions because the time of computation is bounded by the length of the longest path among the shortest paths.

For the case of the *diffusion schedule* (namely for the first order schemes that we refer to here) it is shown that the optimal scheme (i.e. the optimal diffusion matrix) can be determined in a time polynomial in the number of nodes and logarithmic in the optimality parameter [20].

In several papers published recently we can find *extensive* modifications of original ant algorithms where the capabilities of the single ant (agent) are increased enormously. For example, ants can travel back toward their source (see, e.g., [39]) after they have visited the destination. This algorithms obviously have a little difference if any from the centralised or slightly decentralised algorithms for the computation of shortest paths used for the routing optimization of communication and computation networks (see e.g. [7]). In this case the ants are the messages that deliver information about the local states of the network to the central controller (controllers), which in their turn recalculate the matrices of the shortest paths.

7 Acknowledgement

This work is supported by the External Research Project, Hewlett-Packard Labs, Bristol, UK.

References

[1] Adamatzky A. Neural algorithm for constructing minimum spanning tree of a finite planar set *Neural Networks World* **6** (1991) 335-339.

[2] Adamatzky A. and Tolmachiev D. Chemical processor for computation of Voronoi diagram *Adv. Mater. Optics Electr.* **6** (1996) 191-196.

[3] Adamatzky A. Computation of shortest path in cellular automata *Mathl. Comput. Modelling* **23** (1996) 105-113.

[4] Adamatzky A. and Holland O. Voronoi-like nondeterministic partition of a lattice by collectives of finite automata *Mathl. Comput. Modelling* **28** (1998) 73-93.

[5] Adamatzky A. and Holland O. Swarm intelligence: representations and algorithms In *Proc. Int. Workshop on Distributed Artificial Intelligence and Multi-Agent Systems* (St. Petersburg, Russia, 1997) 13-22.

[6] Adamatzky A. and Holland O. Electricity, chemicals, ants, and agents: a spectrum of swarm based techniques (1998), http://www.uwe.ac.uk/facults/eng/ias/andrew/ANT98.HTM.

[7] Ahuja M. and Zhu Y. A distributed algorithm for minimum weight spanning tree based on echo algorithms In: *Proc. Int. Conf. Distr. Computing Syst.*, 1989, 2-8.

[8] Aldous D. J. The random walk construction of uniform spanning trees and uniform labelled trees *SIAM J. Disc. Math.* **3** (1990) 450-465.

[9] S. Appleby and S. Steward, Mobile agents for control in telecommunications networks, *British Telecom Technology Journal* **12**, 104-113 (1994).

[10] R. Beckers, O.E. Holland and J.L. Deneuborg, From local actions to global tasks: stigmergy and collective robotics. In: *ALIFE IV: Proc. 4th Int. Workshop on the Synthesis and Stimulation of Living Systems*, Eds. R.A. Brooks and P. Maes (MIT Press, 1994), 181-189.

[11] Bonabeau E., Henaux F., Guerin S., Snyers D., Kuntz P. and Theraukaz G. Routing in telecommunication networks with "smart" ant-like agents *SFI Research Paper*, 1998.

[12] Broder A. Generating random spanning trees, In: *Proc. Symp. Foundations of Computer Sci.* (New Yors: IEEE Press, 1989), 442-447.

[13] Butrimenko A. V. On the search for optimal routes in changing graphs *Izv. Akad. Nauk SSSR. Ser. Tekhn. Kibern.* **6** (1964).

[14] Chin F. and Ting H.F. Improving the time complexity of message-optimal distributed algorithm for minimum-weight spanning tree *SIAM J. Comput.* 19 (1990) 612-626.

[15] F. Chong, Analog techniques for adaptive routing on interconnection networks, *M.I.T. Transit Note No. 14*, 1993.

[16] A. Colorni, M. Dorigo and V. Maniezzo, Distributed optimization by ant colonies, *Proc. of Europ. Conf. on Artificial Life.*, Eds. F. Varela F. and P. Bourgine, 134-142 (1991).

[17] Cook S. and Rackoff C. Space lower bounds for maze threadability on restricted machines *SIAM J Comput* **9** (1980) 636-652.

[18] Cybenko G. Load balancing for distribute memory multiprocessors, *J. Parallel Distr. Comput.* **7** (1989) 279-301.

[19] Dantzig B.B. On the shortest route through a network *Management Science* **6** (1960) 187-190.

[20] Diekmann R., Muthukrishnan S. and Nayakkankuppam V. Engineering diffusive load balancing algorithms using experiments. In: *Proc. IRREGULAR'97, Lecture Notes in Computer Science* 1253 (1997) 111-122.

[21] Diekmann R., Monien B. and Preis R. Load balancing strategies for distributed memory machines, *Computer Science Technical Report Series "SFB"* No tr-rsfb-97-050, University of Padeborn, 1997.

[22] M. Dorigo, V. Maniezzo and A. Colorni, The Ant System: optimization by a colony of cooperating agents, *IEEE Trans. Syst. Man Cybern.* **26**, 1-13 (1996).

[23] Doyle P. .and Snell J.L. *Random Walks and Electrical Networks* (Washington: Mathematical Association of America, 1984).

[24] Helbing D., Molnar P. and Schweitzer F. Computer simulations of pedestrians dynamics and trail formation (1998)

[25] Gallager R.G., Humblet P.A. and Spira P.M. A distributed algorithm for minimum-weight spanning tree *ACM Tranbs. Programming Languages and Systems* **5** (1983) 66-77.

[26] L.M. Gambardella and M. Dorigo, Ant-Q: A reinforecement learning approach to the traveling salesman problem, *Proc. of the 12th Int. Conf. on Machine Learning*, 252-260 (1995).

[27] Subramanian D., Druschel P. and Chen J. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proc. IJCAI-97: Int. Joint Conf. AI* (Morgan Kaufmann, 1997) 832-838.

[28] R. Lyons and Y. Peres *Probability on Trees and Networks* (a book in progress), 1997, http://www.ma.huji.ac.il/~peres/index.html.

[29] Littman M. and Boyan J. A distributed reinforcement learning scheme for network routing *Carnegi Mellon University Computer Science Report CMU-CS-93-165*, 1993.

[30] G.F. Marshall and L. Tarassenko, Robot path planning using VLSI resistive grids, *Proc. 3rd Int. Conf. Artificial Neural Networks* (Brighton, UK, 1993), 163-167.

[31] A. Ripoll, M.A. Senar, A. Cortes and E. Luque Mapping and dynamic load-balancing strategies for parallel programming *Computers and Artificial Intelligence* 17 (1988) 481-491.

[32] R. Schoonderwoerd, O. Holland, J. Bruten and L. Rothkratz, Ant-based load balancing in telecommunication networks, *Adaptive Behaviour* **5**,169-207 (1996).

[33] R. Schoonderwoerd, O. Holland and J. Bruten, Ant-like agents for load balancing in telecommunication networks, *Proc. 1st Int. Conf. On Autonomous Agents*, Marina del Rey, USA, 209-216, 1997.

[34] Stefanyuk V. L. On mutual assistance in the collective of radio stations *Information Transmission Problems* **7** (1971) 103-107.

[35] Stefanuk V. L. From multi-agent systems to collective behaviour In: *Proc. of the Int. Workshop Distributed Artificial Intelligence and Multi-Agent Systems* (St. Petersburg, Russia, 1997) 223-224, 327-338.

[36] Takaoka T. An efficient parallel algorithm for the all pairs shortest path problem (1991).

[37] M.L. Tsetlin, *Automaton Theory and Modelling of Biological Systems* (New York: Academic Press, 1973).

[38] A. Vergis, K. Steiglitz and B. Dickinson, The complexity of analog computation, *Mathematics and Computers in Simulation* **28** (1986) 91-113.

[39] White T., Pagurek B. and Oppacher F. Connection management using adaptive mobile agents In: *Proc. Int. Conf. Parallel Distributed Processing Techniques and Applications* (CSREA Press, 1998) 802-809.

[40] Xu C.-Z. and Lau F.C. Optimal parameters for load balancing with the diffusion method in mesh networks, *Parallel Processing Letters* (1998).

[41] Xu C. Monien B., Löling R. and Lau F. Nearest-neighbour algorithms for load-balancing in parallel computers, *Concurrency - Practice and Experience* **7** (1995) 707-736.

AGENT VIRTUAL ORGANIZATIONS WITHIN THE FRAMEWORK OF NETWORK COMPUTING: A CASE STUDY *

Stanisław Ambroszkiewicz
Institute of Computer Science, Polish Academy of Sciences
al. Ordona 21, 01-237 Warsaw, Poland
email: sambrosz@ipipan.waw.pl, http://www.ipipan.waw.pl/mas/

Abstract. *We study the concept of agent virtual organization [16, 23] and show how it relates to the paradigm of Network Based Computing [30]. We also discuss the paradigm of BDI-agent trying to show that sophisticated architecture of BDI-agent can not be efficiently applied for large worlds. As the working example of virtual organization we consider a model of virtual enterprise.*

Key words: *Agent virtual organization, agent-based manufacturing, virtual enterprise formation.*

1 Introduction

Autonomous, adaptive and cooperative software mobile agents are well suited for domains that require constant adaptation to changing distributed environment or changing demands. Actually cyberspace and manufacturing enterprise are such domains so that there is increasing interest in applying agent technologies there.

Cyberspace, in the shape of the Internet, intranets, and the World Wide Web, has grown phenomenally in recent years. Cyberspace now contains enormous amounts of information and is also being increasingly exploited for a number of business and other applications. Software agents can act as smart personal assistants, roam cyberspace to collect required information on behalf of the users and conduct a variety of business transactions and functions online.

Taking the new network technology as granted a new manufacturing strategy have been proposed recently: Agile Virtual Enterprise (AVE) [16, 23, 34] as a mechanism for enabling the industry to achieve increased agility (the ability to thrive on frequent and unexpected change). An Agile Virtual Enterprise is a rapidly configured, multi-disciplinary network of firms organized to design and produce a specific product. It is "virtual" because it relaxes the conventional restriction that an enterprise be a single legal entity, headquartered in a single place, with close synchronization among its various functions.

So that there is a need for the software agents to be able to collectively perform complex, collaborative tasks. It is clear that a single agent alone can not perform efficiently its tasks. For large open worlds such as cyberspace simple cooperation mechanisms like, like contracts [38], exchange of resources and knowledge, signing a draft [5], and role/task delegations [11], are not sufficient, especially if the agents' goals are interrelated in a way that fulfilling the goal of a single agent must involve a cooperation of a large number of agents. In that case agent organizations seem to be a panacea to manage the necessary cooperative work.

However, the problem is with the concept of organization as well as with the understanding and modeling organization formation mechanisms, see [31, 32, 35] for an overview.

In the context of Multi-Agent Systems, organizations are regarded as an important aspect of agent coordination, see [27]. However, here the organizations are understood as static structures created by the the designer of the system [13] where agents are designated specific roles to undertake. Another approach is based on so called "social laws" [40, 41, 33] where a society of agents adopts a set of laws (like road traffic rules) that constrain or specify individual agent behavior, see [18] for an application.

According to the prevailing paradigm in current multi-agent systems research, organizations emerge from patters of agents interactions, so that the work is rather centered on agent architectures than on agent organizations. The concepts of organization and organization formation as advanced cooperation mechanisms, although regarded as important, were not so far a subject of extensive research, modeling and application in MAS, an exception is [19, 25] and recently [20].

Manufacturing in a large distributed world seems to be a perfect environment for modeling virtual organizations; in this case virtual enterprises. However the

*Supported by the State Committee for Scientific Research under the grant No. 8 T11C 008 12

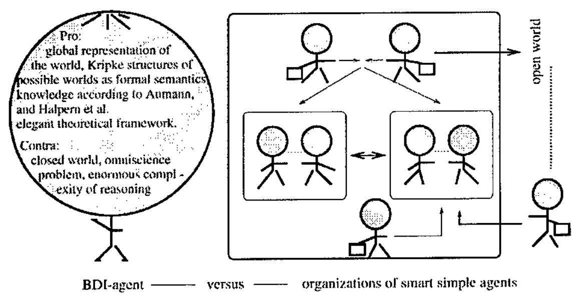

Pro:
global representation of the world, Kripke structures of possible worlds as formal semantics
knowledge according to Aumann, and Halpern et al.
elegant theoretical framework.
Contra:
closed world, omniscience problem, enormous compl - exity of reasoning
open world
BDI-agent —— versus —— organizations of smart simple agents

Figure 1: Shifting of agent paradigm: from sophisticated BDI-agent to simple smart agents in organizations.

manufacturing is so complex that we have decided to construct an abstract model of the environment. So that we propose a simple model of "a networked world" represented by a computer network where workstations play roles of computing factories with specific inputs and outputs.

Thus our research objective consists in modeling agent virtual organizations. For this purpose we choose manufacturing enterprise as the working example. Our goal consists in constructing generic formation mechanisms of efficient enterprises that manufacture the products specified by a demand.

The paper is organized as follows. In Section 2 we present our methodology of agent virtual organizations. Section 3 is devoted to introduction of our abstract model of manufacturing. In Section 4 we present our methodology of virtual enterprise formation. In Section 5 we outline some details of our implementation of the concepts of enterprise and reconfiguration mechanism. Section 6 contains a discussion whereas Section 7 a short conclusion.

2 Agent Virtual Organizations

The idea of virtual organization is not novel and has been exploited for several years, see [9]. We have rediscovered this idea during implementation of the concept of BDI-agent (see [10, 36]) for environments like the one shown in Fig. 2. In the case of small worlds the BDI-agents perform their job well. However if the world is getting larger, then sophisticated architectures of BDI-agents become not efficient. The reason for that is that if agent has to much knowledge about the world and tries to compute good (cooperative) plan to realize its goal, then it takes some time for him to do so. If the world is large and there are a lot of agents that may help or prevent the agent to achieve its goal, computation of a good plan takes so long time that after completing the computation, the plan becomes out-of-

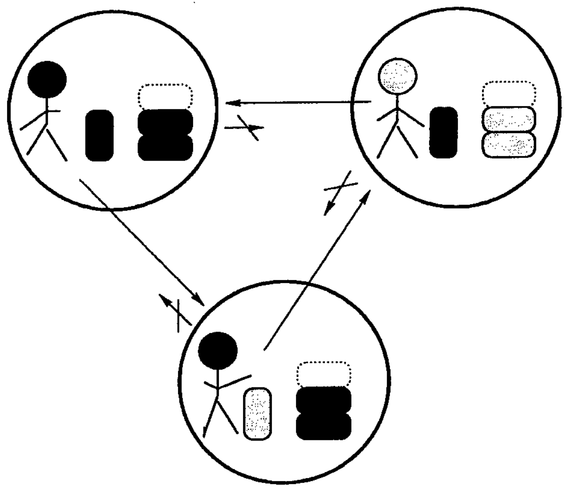


Figure 2: Simple agent world: Three agents; each of them is collecting blocks of the same color as its own head. For that simple case BDI agent works well.

date, because the world has changed in the meantime. Our implementation experiences forces us to change the concept of agent architecture, see Fig. 1. In order to be efficient the agent must be simple. However this very simplicity implies that the agent alone can not realize its goal. To manage this problem we came up with the idea of team [4]. It turned out that our teams are simple examples of agent virtual organization.

Having these examples we started to study the general idea of agent virtual organizations (AVO for short). Looking over the literature we have realized that the idea of AVO could have application in so called Network-Based Programming.

2.1 Network-Based Programming

Computer networks offer new application scenarios that cannot be realized on single workstations. Resources and services may be distributed because of political, organizational, etc. reasons which make a centralization impossible. The parallelism gained through the simultaneous processing of subtasks at distributed sites may lead to a better system throughput. However for large computer networks, the classical programming paradigm (see Fig. 3) in not sufficient. It is hard to imagine and the more to realize the control, synchronization and cooperation for hundreds of processes running on different workstations. It is at least not efficient if not unrealistic at all to program, run and control each application on any workstation separately. Here comes up the idea of (mobile) autonomous agent as a solution to this problem. But certainly this is not enough. To run an application

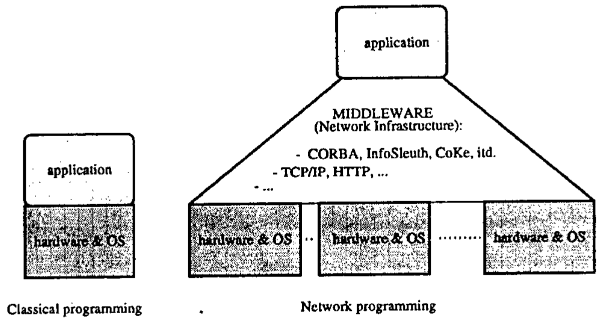


Figure 3: Computing on a single workstation versus network computing.

on a network, an infrastructure is needed for assuring communication and coordination between heterogeneous workstations. That very infrastructure is called "middleware" in the literature. CORBA [14], CoKe [30], InfoSleuth [24], and our Agent Platform [2] are examples of such middleware. Hence, the Network-Based Programming paradigm [30] goes far beyond the classical paradigm of distributed computing where each process is run separately and the communication between the processes is already given.

The new paradigm postulates design and implementation of interaction (cooperation) mechanisms for autonomous software agents because the network infrastructure restricted only to CORBA or CoKe is not sufficient. So that there are several attempts (see FIPA [21] for example) to extend network infrastructure by introducing standard services and facilitators that can be used by software agents (see the Fig. 4.).

The concept of Dynamic Interaction Model (see [37]) is closely related to the idea of dynamic interaction mechanisms shown in Fig. 4

Once we grasp the idea of Fig. 4, there is only one small step to the concept of virtual organizations seen as advanced agent interaction mechanisms. In this context agent-based programming consists in not only designing and implementing agent architectures but also in designing and implementing virtual organizations for the agents.

2.2 Realization of the idea of AVO

The idea of agent virtual organization seems to be clear, however the real problem is with implementation of this idea. There are several research project that aim at understanding the concept of organization, see [35] as a quite recent source of references.

The problem is that even simple organization has sophisticated structure, in which we can distinguish: the goal of the organization, business process realizing that

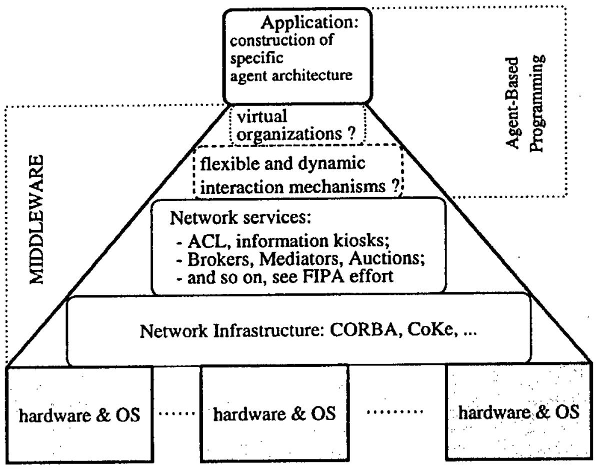


Figure 4: Agent virtual organization within the framework of network computing.

goal, internal information flow, and organizational dependencies concerning the decision power distribution. If all this sophisticated structure must be incorporated into architecture of simple agent (see Fig. 5), then the question arises whether it is possible. It is already proved in [4] that it is possible for simple cases. In constructing our model of virtual enterprise we struggle to prove that it is also possible for a real complex case.

3 Our working model: virtual enterprise

We propose a simple model of "a networked world" represented by a computer network where workstations play roles of computing factories with specific inputs and outputs. Each computing factory is managed by one agent called *factory agent*. There is a global demand for some final products (computations) so that in order to manufacture one of such final products the factories must be formed in several competing supply chains (networks). However, if the world is large and production technologies are complex, then supply chain formation alone is not sufficient. The supply chain must be maintained, integrated, and reconfigured to be able to compete with other chains. For this purpose the factories of a supply chain are forced to form a joint enterprise at least for some period of time.

In our model it is supposed that all this work of formation, integration and reconfiguration is done by autonomous mobile agents created by the factory agents. To formalize the model let us consider graph $G = (V, E)$ representing a network of servers, where V is

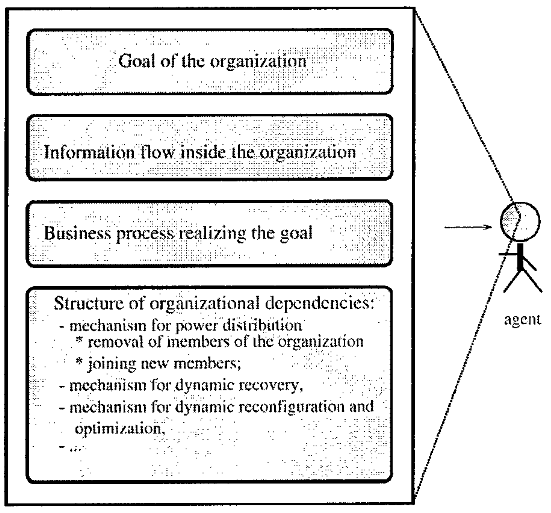

Goal of the organization
Information flow inside the organization
Business process realizing the goal
Structure of organizational dependencies:
- mechanism for power distribution
* removal of members of the organization
* joining new members;
- mechanism for dynamic recovery,
- mechanism for dynamic reconfiguration and optimization,
- ...
agent
Organization

Figure 5: The problem: How to build the generic organization structure into agent architecture ?

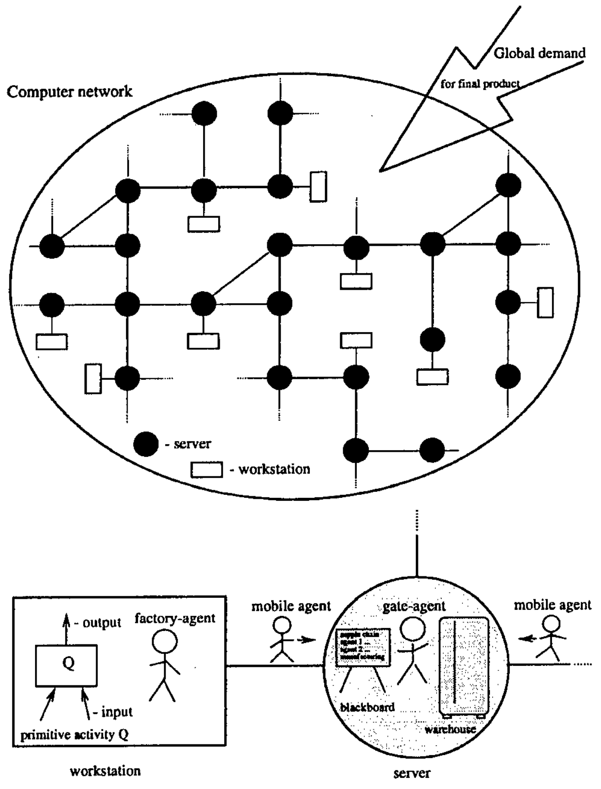


Figure 6: The model.

the set of vertices (servers), $E \subseteq V \times V$ is the set of edges (connections), see Fig. 6.
There is one static-gate-agent at each server v, denoted also by v, responsible for management on that server. The management consists in:
1. Maintaining a blackboard for information exchange by the mobile agents.
2. Storing the products manufactured by factories.
3. Transportation of mobile agents and products to other servers in the neighborhood.
The gate-agents are supposed to be fully obedient. Around any server there are workstations connected only to this server. Let the set of all workstations be denoted by W. There is one **computing factory** residing on any workstation $w \in W$ (denoted also by w). A computing factory means here a specific hardware & software for specific computations. Any such computing factory has precisely specified (multiple) input and (multiple) output, so that it may be represented as an operation of the following form:

$$Q_w : \prod_{i \in In_w} A_i \to \prod_{j \in Out_w} B_j,$$

where A_i, $i \in In_w$ corresponds to the multiple input, whereas B_j, $j \in Out_w$ to the multiple output. Similar notions of operation are frequently used in the workflow literature, for example the notion of *activity* and *task* in [1, 12]. Our model may be seen as quite generic if we suppose that operations are primitive workflow activities.
A computing factory is managed by one autonomous agent called *factory agent.* His goal consists in getting maximal and firm profit form selling the manufactured products. It is supposed that any factory w is characterized by so called production stream S_w that describes how much of the input, "time" and "energy" is needed to produce a particular amount of the output. Hence, the production stream is interpreted as the time, energy, and resource complexity of the production technology used in the factory. Input (respectively output) may be interpreted as types of products, commodities, resources needed for production (respectively as types of final products of the factory). Commodities produced or needed for production may be stored at the servers, however there is a fixed fee for transportation, and storing a unit of product per a unit of time, as well as for writing or reading an info on the blackboard.
It is supposed that message passing as well as transportation of products, and energy (money) is done by mobile agents; each transportation from one server to another costs a fixed fee. Each mobile agent belongs to its master, i.e. a factory agent (enterprise agent) that has created it and equipped it with energy (money) needed for travel, product transportation, message passing, etc. over the network of servers. The creation of a mobile agent may be also a subject of paying a fixed fee.
Energy (money) is acquired from selling products. The

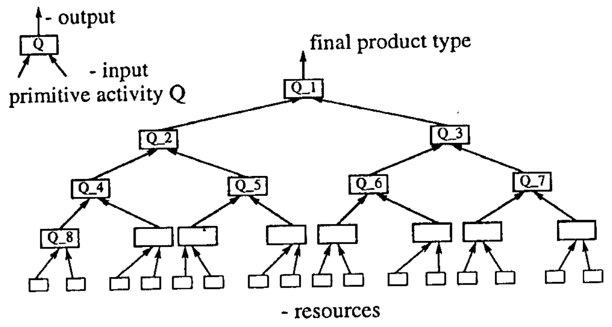


Figure 7: A technology process of the final product.

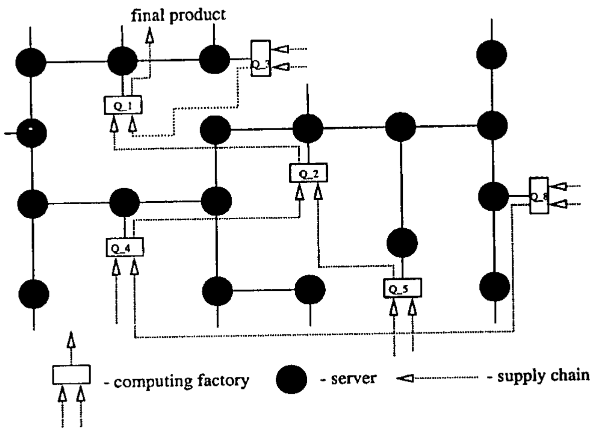


Figure 8: A fragment of supply chain corresponding to the technology from Fig. 7.

source of money is the global demand for the final products. The demand is external and is determined by the designer of the system.

One fixed server is the place for auction where the designer collects the proposals from already formed enterprises with already constructed technology process (see Fig. 7) and corresponding complete supply chains (see Fig. 8). Generally, the supply chain is a network of suppliers, factories, warehouses, distribution centers and retailers trough which raw materials are acquired, transformed and delivered to the customer. The completeness of supply chain means here that it contains all intermediate factories (starting from resources) needed to manufacture the final product, with determined supply routes between them in the network.

The proposal of an enterprise specifies delivery time, price and amount of the final product. After collecting all proposals, the designer decides, on the basis of the fixed mechanism, denoted by DM, how to distribute the global demand between the enterprises. It is supposed that the mechanism prefers larger product streams (amounts) and lower prices.

Let us notice that a factory agent can get a payoff only if it is a member of a complete supply chain. The agents have to cooperate with some agents whereas with some others they have to compete. So that in order to realize his goal the agent tries to form (or to join to) a supply chain that can be completed and can assure, in this way, a good profit for him. However, how to do so? Our approach towards solution of the problem relies on constructing simple protocols for the agents to follow, like: *Try to join to a forming (according to a fixed mechanism) enterprise and submit to the all resulting obligation.*

4 Methodology

The crucial point of the methodology applied in our work is the separation of enterprise formation mechanism from the agent architectures.

4.1 Agents

A factory agent tries to form or to join to an enterprise according to a formation mechanism. It is intuitively clear that an incomplete enterprise would accept a factory agent as its member if the resources and/or production capabilities of the agent are needed in the supply chain of the enterprise. Our concept of formation mechanism relies on so called *roles* or positions to be taken by the members of the forming enterprise. The roles are inherent parts of the formation mechanisms. Any agent tries to get a role (position) in an already existing enterprise or in a new one that will assure him maximal and firm profit. In order to find out such forming enterprise and negotiate a profitable role, the agent creates and sends mobile agents. Once the factory agent joins to an enterprise and has got a role, he commits to all obligations prescribed to that role. The obligations consist in fulfilling a more or less specific knowledge-based protocol consistent with the goal of the enterprise. Since the protocol is knowledge-based, the agent is obliged to acquire knowledge from the environment via the mobile agents.
Hence, the whole strategy of a factory agent consists in finding out an opportunity to join to a forming enterprise and negotiate an advantageous position to undertake in that enterprise. What the agent does next is determined by knowledge-based protocol prescribed to the position, so that the agent need not "think" too much about it.

4.2 A concept of enterprise

Enterprise structure consists of three basic and interrelated frames: business process frame, organization frame, and information flow frame.

Business process frame. Business process frame may be viewed as a collection of factories formed into a supply chain together with a management of the supply chain.

Organizational frame. This frame consists of roles and organizational interdependencies between them as well as several interrelated mechanisms:

1. Mechanism for distribution of the net profit of the enterprise among its members.
2. Mechanism for distribution of decision power concerning the status of the enterprise, i.e. who and under what circumstances has the power to:
- remove a factory from the supply chain of the enterprise in favor of a more efficient one,
- join a new factory or another enterprise, and determine a position to the new member.
3. Mechanism for recovery from failures in functioning of supply chain and organizational dependencies.
4. Mechanism for redesign and reconfiguration of technology process and supply chain.

Another important topic here is hierarchical enterprise structure and partial autonomy of enterprise members, i.e. in a formation process of small enterprises into larger one, a small enterprise may preserve a part of its autonomy.

Information flow frame. This frame is to serve for maintaining organizational integrity of the enterprise as well as for optimizing supply chain performance.

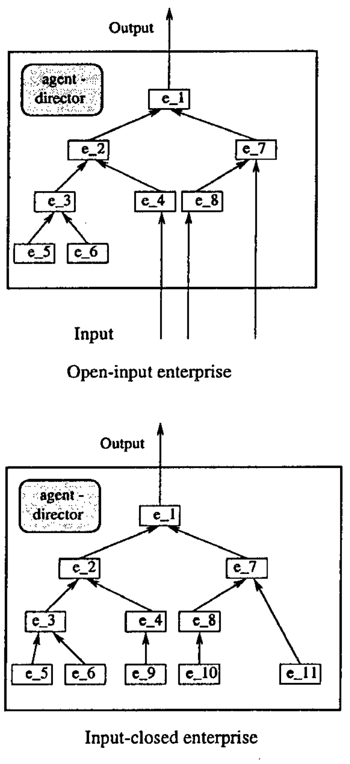


Figure 9: The concept of enterprise.

5 Implementation of the enterprise concept

Our implementation of enterprise concept is extremely simple. It consists of *managing director*, a collection of factories or small enterprises formed into a supply chain with distinguished final products to be manufactured. The basic frames are built into the role of managing director. That is, a mobile agent undertaking that role performs all tasks specified in these three frames. The managing director of an enterprise is equipped with visiting cycle and decision making protocol.

The visiting cycle determines the order of visits the director is obliged to pay to the factories belonging to the enterprise. This constitutes the information flow frame.

For the sake of prices calculation and applying MRP II (Manufacturing Resource Planning) for management we consider only input-closed enterprises (see Fig. 9). This means that the unfolded supply chain of the enterprise contains all intermediate factories (starting from the resources) needed to produce the product manufactured by the enterprise. More details concerning this issue are given in the next section.

In order to introduce the hierarchy to the enterprise structure, we allow the factories belonging to an enterprise to be also enterprises. So that in this way the structure of enterprise becomes recursively nested.

As a mechanism for net profit distribution we use a simple method that divides the profit proportionally to the work load. The method allows to avoid agents' negotiations during enterprise formation process.

Decision making protocol. The goal designated to managing director is to manage the enterprise in a way that would assure maximal production stream. To achieve this goal the managing director is responsible first of all for the supply chain management to produce efficiently desired quantities of the final products. This is done by applying *MRP II* method, see [44]. The second duty of the director is to take decisions (according to the specified protocol) concerning the status of his enterprise. This concerns enterprise reconfiguration to produce more efficiently as well as forming a complete

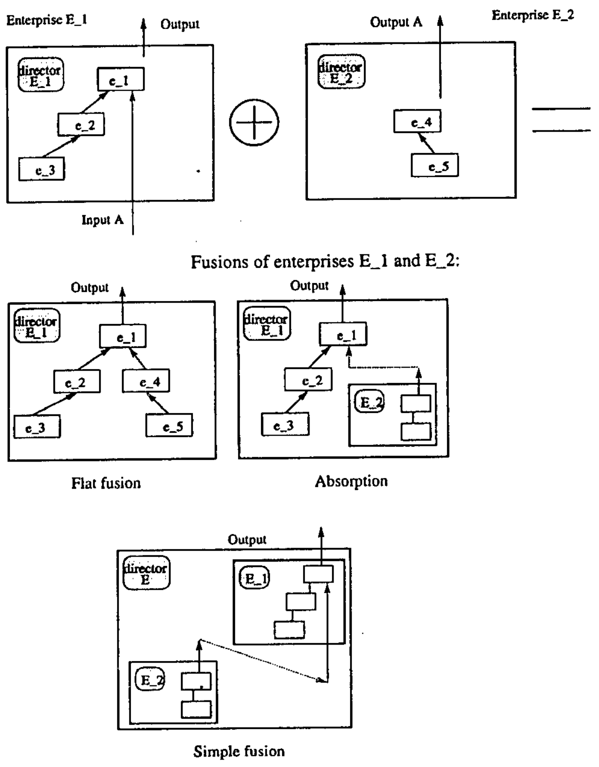


Figure 10: Possible fusion types of two enterprises E_1 and E_2.

supply chain. He may remove a factory from the enterprise, join a factory to the enterprise or make a fusion with another enterprise. His decisions are based on information acquired from the factory agents belonging to the enterprise.

It is important to notice that business, information flow, and organizational frame of the enterprise structure presented above are integrated into one role of managing director. Since we allow the enterprise structure to be hierarchical, also the business, information flow, and organizational frames are structured in a sophisticated hierarchical way. This solution follows our idea of team formation [4].

Now we are going to present some details of the protocol concerning reconfigurations, that is, fusion of two enterprises (see Fig. 10), and removal of a factory from an enterprise (see Fig. 11). We distinguish three types of fusion of two enterprises E_1, E_2 into new one E:

- **Simple fusion.** The enterprises E_1, E_2 preserve their internal structures and become "factories" of the new enterprise E. A mobile agent undertaking the role of managing director of E is created.

- **Absorption.** Enterprise E_2 is joint to E_1 as a new "factory" of the enterprise E_1. The factory E_2 preserves its internal structure.

- **Flat fusion.** Both enterprises E_1 and E_2 loose their internal structure. So that the new enterprise E consists of all factories of E_1 and E_2. One of the managing directors of E_1 and E_2 becomes the managing director of E whereas the other is destroyed.

It is clear that extreme enterprise structures resulting from applying only one fusion type can not be efficient. It seems that a tradeoff between the size of enterprise and the number of hierarchy levels should be found out.

As to the removal of a sub-enterprise e from an enterprise E we distinguish two cases:

1. The managing director wants to remove e.
2. The sub-enterprise e wants to leave the enterprise E.

We consider three scenarios that solve the removal problem:

- **Mutual agreement.** The director of E and the director of e do agree on this subject.

- **Voting.** The removal (leaving) of e is a subject of voting of the directors of all coordinate sub-enterprises.

- **Termination after a fixed period of time.**

6 Related work and discussion

As far as we know our approach to enterprise formation is relatively new one. There are several attempts to apply software agents to management (see [39] for an overview), however the problem of enterprise formation and integration is considered as hard, so that the basic research is focused rather on building universal platforms than on constructing formation mechanisms. One of commercial platforms for management of heterogeneous information (that is actually a basis for efficient manufacturing as well as for workflow management [1, 12, 17]) is InfoSleuth [24]. It seems that the InfoSleuth architecture with user agents, broker agents, and resource agents supports worlds where the service technology is relatively simple, that is, the path between a user who wants a service, agents that perform that service, and resources needed is short. In that case contracts between the agents suffice to perform the service efficiently. However in the case of large worlds with sophisticated service technology, agent organizations seems to be at least of interest as a solution to the problem of effective service performance. Of course, the same concerns manufacturing. Actually

our metaphor of "computing factory" is much more close to the service than to real manufacturing. However, the problems with management, maintenance, recovery, and optimization are similar.
The organizations make the world smaller in the sense that one organization may aggregate tens or hundreds of factories, so that technology processes become simpler although not necessary optimal.
Although agent organizations may be seen as panacea to solve the problem of complex cooperative multi-agent work, simple mechanism like contracts, barters, and markets should not be eliminated.
Our concept of global enterprise containing a complete supply chain (i.e. all intermediate factories starting from resources needed to manufacture final products) is not a perfect solution. It seems that too large enterprises can not be efficient. However, in order to have clear and simple formation mechanism, we consider only input-closed enterprises. For these enterprises the price calculation as well MRP II method can be easily applied. Open-input enterprises can function if contracts or/and markets are available. After completing testing with input-closed enterprise concept we are going to introduce contracts, auctions, and markets to our model.
Another problem is with agents' negotiations. Since it is supposed that the factory agents are interested in maximizing his own profit, it is clear that each agent wants to purchase the input products at lowest prices whereas to sell his output products at highest prices. The negotiations concerning selling–buying are hard problems (see [29, 45]), so that in order to avoid them we introduce (see [7]) a uniform distribution protocol of net profit proportional to load of work performed by enterprise members. In a future work we are going to introduce more flexible negotiation protocols.
A serious drawback of our enterprise concept implementation is the assumption of perfect reliability of enterprise functioning, i.e. no random failure or damage can occur, so that a mechanism for recovery is not needed. The introduction of random failures to our model is a subject of the future work.

7 Preliminary conclusions

The paper reports the work in progress. We have presented our idea of agent virtual organization. We have outlined some details of our working model, methodology applied, agent architectures, and enterprise formation mechanisms. The model presented in Section 2 is already implemented in JAVA on the Agent Platform [2]. Now we are constructing and testing simple cases of agent architectures and formation mechanisms for

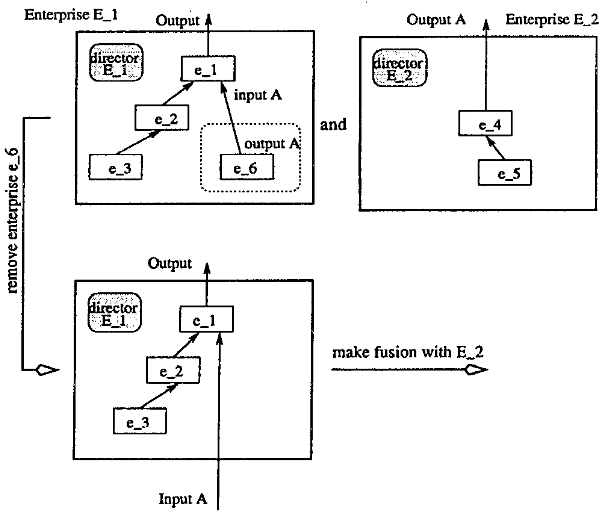


Figure 11: A reconfiguration of enterprise E_1.

small worlds. However, our approach is supposed to be for large worlds, so that performances of our tests can not be representative. We need a testbed for comparison, that is, a universal world model that can be scaled up.
The progress and results of experiments will be published in the subsequent papers.

References

[1] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan. Functionality and Limitations of Current Workflow Management Systems. To appear in IEEE-Expert, special issue on Cooperative Information Systems, 1997

[2] Agent Platform developed at Institute of Computer Science, Polish Academy of Sciences: www.ipipan.waw.pl/mas/

[3] S. Ambroszkiewicz, and J. Komar. A model of BDI-agent in game-theoretic framework. In *Proc. ModelAge-97,* to appear in Springer LNAI. http://www.ipipan.waw.pl/mas/

[4] S. Ambroszkiewicz, O. Matyja, and W. Penczek. Team Formation by Self-Interested Mobile Agents. In Chengqi Zhang and Dickson Lukose (Eds.), *Proc. 4-th Australian DAI-Workshop,* Brisbane, Australia, July 13, 1998. Published in Springer LNAI 1544. http://www.ipipan.waw.pl/mas/

[5] S. Ambroszkiewicz, O. Matyja, and W. Penczek. Cooperation Mechanisms in a Multi-Agent Distributed Environment. In *Proc. ECAI-98 Workshop w8,* Brighton, UK, August 24-28, 1998.

[6] S. Ambroszkiewicz, K. Cetnarowicz, T. Nowak, and B. Radko. Modelling Enterprise Formation and Integration as a Distributed Computing Performed by Mobile

Agents. To appear in Proc. of HUNABC'98, First Hungarian National Conference on Agent Based Computing, May 29-31 (Fri-Sun) 1998, Budapest, Hungary.

[7] S. Ambroszkiewicz, K. Cetnarowicz, and B. Radko. Enterprise formation mechanism based on mobile agents. In Proc. of Intelligent Agents in Information and Process Management. A. Holsten, G. Joeris, C. Klauck, M. Klush, H.-Mueller, and J.P. Mueller (Eds.) Workshop at the 22nd German Annual Conference on Artificial Intelligence in Bremen (KI-98), TZI-Bericht Nr. 9, 1998. pp. 23-34.

[8] S. Ambroszkiewicz, K. Cetnarowicz, O. Matyja, and B. Radko. Modeling Virtual Enterprise: agent-based approach. In Proc. Multi Agent Systems Models Architecture and Applications. F. J. Garijo, and Ch. Lemaitre (Eds.), II Iberoamerican Workshop on D.A.I and M.A.S., October 1-2 1998 Toledo, Spain.

[9] Virtual Organizations research project sponsered by the Institute of Information Systems, Department of Information Management at the University of Berne. www.virtual-organization.net/ Æ

[10] M. E. Bratman. *Intentions, Plans, and Practical Reason.*

[11] C. Castelfranchi and R. Falcone. From Task Delegation to Role Delegation. In Proc. AI*IA-97, LNAI 1321, 278-289.

[12] A. Cichocki, A. Heal, M.Rusinkiewicz, and D. Woelk. *Workflow and Process Automation: Concepts and Thechnology.* Kluwer Academic Publishers, 1998.

[13] D. Cockburn and N.J.Jennings. ACHRON: A Distributed Artifical Intelligence System for Industrial Applications. In G. M. P. O'Hare and N. R. Jennings (eds.): "Foundations of Distributed Artificial Intelligence, Sixth Generation Computer Technology Series, John Wiley & Sons, New York, 1996.

[14] The Object Management Group's Common Object Request Broker Architecture (OMG/CORBA) http://www.acl.lanl.gov/CORBA/

[15] P.R. Cohen, H.J. Levesque, and Ira Smith. On Team Formation. In G. H. Hintikka and R. Tuomela (eds.), *Contemporary Action Theory,* Volume 2: Social Action, Synthese Library, Kluwer Academic Publishers, Dordrecht, Netherlands, 1997, pp. 87-114.

[16] W. Davidow, and M. Malone. *The virtual corporation: structuring and revitalizing the corporation for the 21st century.* - New York: Harper Business, 1992.

[17] J. Debenham. An Experimental Agent-based Workflow System. In Proc. PAAM'98: The Practical Application of Intelligent Agents and Multi-Agents, March 1998, London UK

[18] K. Decker. Distributed Artificial Intelligence Testbeds. In G. M. P. O'Hare and N. R. Jennings (eds.): "Foundations of Distributed Artificial Intelligence , Sixth Generation Computer Technology Series, John Wiley & Sons, New York, 1996.

[19] Y. Demazeau and A.C. R. Costa "Populations and Organisations in Open Multi-Agent Systems", 1st Symposium on Parallel and Distributed AI, Hyderabad, India, July 1996.

[20] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In Proc. ICMAS-98.

[21] The Foundation for Intelligent Physical Agents (FIPA) http://drogo.cselt.stet.it/fipa/

[22] M.S. Fox et al. An Organizational Ontology for Enterprise Modeling. In M. Prietula et al.(Eds.). Simulating Organizations. pp. 131-152 AAAI Press and MIT Press, 1998.

[23] S. Goldman, R. Nagel, K. Preiss. *Agile competitors and Virtual Organizations: Strategies for Enriching the Customer.* - New York: Van Nostrand Reinhold, 1995.

[24] InfoSleuth project at MCC, http://www.mcc.com/

[25] T. Ishida, M. Yokoo, and L. Gasser. An organizational Approach to Adaptive Production Systems. In *Proc. of 8th National Conf. on Artificial Intelligence,* Boston, USA, 1990.

[26] St. Jablonski, and Ch. Bussler. *Workflow Management - Modeling Concepts, Architectures, and Implementations.* International Thomson Computer Press, London, 1996.

[27] N. R. Jennings. Coordination Techniques for Distributed Artificial Intelligence. In G. M. P. O'Hare and N. R. Jennings (eds.): "Foundations of Distributed Artificial Intelligence, Sixth Generation Computer Technology Series, John Wiley & Sons, New York, 1996.

[28] D. Kinny, M. Ljungberg, A. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner (Eds.), *Artificial Social Systems, LNAI 830,* 1992.

[29] S. Kraus, and D. Lehmann. Designing and Building a Negotiation Automated Agent. *Computational Intelligence,* 11(1), pp. 132-171, 1995.

[30] E. Kuhn and G. Nozicka. Post / Server Coordination Tools. In W. Conen, G. Neumann (Eds.) Coordination Technologies for Collaborative Applications. LNCS 1364. pp.231-253.

[31] T. W. Malone, K. Crowston, J. Lee, B. Pentland, Ch. Dellarocas, G. Wyner, J. Quimby, Ch. Osborne, amd A. Bernstein. Tools for inventing organizations: Toward a handbook of organizational processes. Center for Coordination Science, MIT, http://ccs.mit/edu/

[32] T. W. Malone and K. Crowstone. Towards an interdisciplinary theory of coordination. *Computing Surveys,* 26(1), 87-119, 1994

[33] S. Ossowski and A. Garcia-Serrano. Social Structure in Artificial Agent Societies. In Proc. ATAL-98.

[34] H. Van Dyke Parunak. Technologies for Virtual Enterprises. Forthcoming in the *Agility Journal* 1997, http://www.iti.org/~van

[35] M. Prietula et al.(Eds.). Simulating Organizations. AAAI Press and MIT Press, 1998.

[36] A. S. Rao and M. P. Georgeff. Modelling rational agents within a BDI–architecture. In R. Fikes and E. Sandewall, editors, *Proc. of the 2rd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pp. 473–484, Cambridge, Mass., April 1991, Morgan Kaufmann.

[37] A. M. Ribeiro, and Y. Demazeau. A Dynamic Interaction Model for Multi-Agent Systems. In Proc. Multi Agent Systems Models Architecture and Applications. F. J. Garijo, and Ch. Lemaitre (Eds.), II Iberoamerican Workshop on D.A.I and M.A.S., October 1-2 1998 Toledo, Spain.

[38] T. Sandholm. Agents in Electronic Commerce: Component Technologies for Automated Negotiations and Coalition Formation. In *Proc. ICMAS'98*, pp. 10-11, Paris 1998.

[39] W. Shen and D. H. Norrie. Agent-Based Approaches for Intelligent Manufacturing: A State-of-the-Art Survey. In *Proc. DAI'98, Fourth Australian Workshop on Distributed Intelligence,* Brisbane, Australia, 13th July 1998, to appear in Springer LNAI.

[40] Y. Shoham and M. Tennenholtz. On the Synthesis of Useful Social Laws for Artificial Agent Societies. In *Proc. of 10th National Conf. on AI,* San Jose, USA, 1992.

[41] J. Sichman, R. Conte, C. Castelfranchi, Y. Demazeau. A social reasoning mechanism based on dependence networks. In *Proc. 11th ECAI,* 1994.

[42] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research,* 7, 1997, pp. 83-124.

[43] J. Tirole. *Industrial Organization.* The MIT Press, 1990

[44] R. Wild. *Production and Operations Management.* Cassell Educational Ltd. 1995

[45] G. Zlotkin and J.S. Rosenschein. Mechanisms for Automated Negotiation in State Oriented Domains. *Journal of Artificial Intelligence Research* 5, pp. 163-238, 1996.

LOCAL INTERACTIONS, EXPLICIT COMMUNICATION AND CAUSAL KNOWLEDGE IN GAMES AND MULTI-AGENT SYSTEMS*

Stanisław Ambroszkiewicz and Wojciech Penczek
Institute of Computer Science, Polish Academy of Sciences
al. Ordona 21, 01-237 Warsaw, Poland
email: sambrosz, penczek@ipipan.waw.pl, http://www.ipipan.waw.pl/mas/

Abstract. *In this paper we suggest new structures for modeling games (multi-agent systems). We use prime event structures as the branching runs representing all the possible plays in games. Event structures have been already successfully applied in the theory of distributed systems [18] and several temporal logics have adopted them as frames. We show that a prime event structure extended with a utility function defined on the terminal nodes can be naturally considered as distributed game in extensive form. The notion of passing knowledge introduces a novel concept to the standard definition of games in extensive form. As well, the notion of information sets (corresponding to agent's knowledge) gets new insight, because now the information sets cannot be defined in an arbitrary way, i.e., they must correspond to causal dependencies in the system.*

Key words: *Multi-agent systems, games, even structures, causality, knowledge.*

1 Introduction

It is intuitively clear that games are closely related to distributed systems and multi-agent systems. However, the classical definition of extensive games takes its inspirations from strategic games like chess, gomoku, mancala. In comparison with economical and war games, these games seem to be artificial and impose several severe restrictions like global states of a play and consecutive order of players' moves. Moreover, in the classical definition of game, the communication, knowledge exchange, and cooperation between the players cannot be defined explicitly.

The essential feature of real economic and war games, that actually may be considered also as multi-agent systems, are local interactions of players (agents). During such interactions, agents may communicate, cooperate, and pass knowledge to each other. So that usually in the real games there is no need for considering global time, global states, and consecutive order of players' moves especially when they are independent. Hence, the classical definition of extensive games should be revised. Our approach may be seen as an attempt towards such a revision. An extension of classical definition can be found in Kaneko & Dubey [6], where the notion of simultaneous moves is introduced.

In this paper we suggest new structures for modeling games. We use prime event structures as the branching runs representing all the possible plays in games. Event structures have been already successfully applied in the theory of distributed systems [18] and several temporal logics have adopted them as frames [12, 13, 14, 10]. We show that a prime event structure extended with a utility function defined on the terminal nodes can be naturally considered as distributed game in extensive form. The term *distributed* corresponds to the distributed local interactions between the players (agents). So that local interaction of two or more players can be explicitly modeled. During such interaction the players have opportunity to communicate or acquire knowledge. However, the lack of global states of a play (resulting from distributed interactions) does not disturb in applying standard techniques of game theory, like backwards induction, selection of rationalizable strategies, or equilibrium points. It seems that this very distributed interactions make the extensive games more realistic and interesting. In order to show that distributed games are proper revision of the classical notion of extensive games, we prove an analog of the second theorem of Kuhn [5, 7].

The notion of passing knowledge introduces a novel concept to the standard definition of games in extensive form. As well, the notion of information sets (corresponding to agent's knowledge) gets new insight, because now the information sets cannot be defined in an arbitrary way, i.e., they must correspond to causal dependencies in the system. For example, an agent cannot know that an event, say e, occurred unless either he

*Partially supported by ESPRIT project No. 20288 CRIT-2 has participated in that event or knowledge about e was acquired from another agent during a meeting. Knowledge acquisition is modeled by introducing queries in the similar manner as in databases [3].

It is our intention that the framework sketched above can serve as a semantics for formal logics of distributed games and multi-agent systems. In the theory of distributed systems, knowledge formulas are usually interpreted over infinite linear or branching runs of the systems [4, 9, 16, 17]. It is clear that capturing changes in state due to actions is crucial for successful modeling of knowledge. While these changes are usually present in the frames, logical formalisms quite rarely incorporate them. One of the reasons is that when actions are incorporated into global state formalisms, this leads to high undecidability [8, 9]. In [15], Penczek considers a temporal logic of causal knowledge interpreted over a variant of flow event structures. The logic is decidable and possesses a complete axiomatization. Therefore, there is an important reason to develop semantic models for multi-agent systems and games, which can be characterized by decidable temporal logics of knowledge incorporating the effects of actions.

Classical definitions of knowledge are built on global states and global time, see Aumann [2], and Halpern et al. [4]. The consequence of that definition is logical omniscience of the players and an arbitrarily deep nesting of knowledge operators, that is, formulas under consideration are of the form: *agent i_1 knows that agent i_2 knows that ... that agent i_n knows that event e occurred.* This very omniscience is frequently regarded as a drawback especially if player is to be modeled to take decisions in real time. Moreover when modeling such a player, it turns out that the representation of the whole game-world must to be put into the "brain" of the player, see the concept of BDI-agent of Rao & Georgeff [17]. This is acceptable if the world is small, say up to ten players and several hundreds of decision nodes (global states), but if the world is getting larger then it is computationally unrealistic do deal with such a model [1]. Hence, if the world is large and/or what is even worse, the world is "open," then the classical notion of knowledge remains only an elegant theoretical notion.

Our alternative proposal to the classical notion of knowledge of Aumann, Halpern et al. is acquisition of knowledge by the agents (initially they may know almost nothing) via communication in environment (possibly "open" one) with local interactions instead of logical omniscience and arbitrary deep nesting of knowledge operator.

The rest of the paper is organized as follows. In section 2 the frames of distributed games are introduced. Causal knowledge and distributed games are defined in section 3. Section 4 contains analogs of Kuhn's theorems. Concluding remarks are given in section 5.

2 Frames of distributed games

We start with a definition of general partially ordered structures, which are frequently used for representing behaviors of distributed systems.

Definition 2.1 (Winskel [18]) *A* labeled prime event structure *(***lpes**, *for short) is a 5-tuple $\mathcal{R} = (E, A, \rightarrow, \#, l)$, where*

1. *E is a finite set, called a set of* events *or* action occurrences,
2. *A is a finite set, called a set of* actions,
3. *$\rightarrow$ is a subset of $E \times E$; it is an irreflexive, acyclic binary relation, called the* immediate causality relation *such that $\downarrow e \overset{def}{=} \{e' \in E \mid e' \rightarrow^* e\}$ is finite for each $e \in E$, where $\rightarrow^*$ is the reflexive and transitive closure of $\rightarrow$,*
4. *$\#$ is a subset of $E \times E$ and is a symmetric, irreflexive relation, called* conflict relation, *such that $(\# \circ \rightarrow^*) \subseteq \#$ (called* conflict preservation condition*),*
5. *The intersection of $\rightarrow^*$ and $\#$ is empty,*
6. *$l : E \longrightarrow A$ is a labeling function.*

We intend to introduce agents (players) in such a way that the lpes's represent behaviors of multi-agent systems or game frames by taking occurrences of agents' actions as the starting point. Every occurrence of an action is modeled as the separate event; a labeling function l indicates for each event which action it is an occurrence of; $l(e) = a$ means that event e is an occurrence of action a. The two relations $\rightarrow, \#$ capture, respectively, the causality, and conflict relationship between events.

We say that two events are in conflict if they cannot occur in the same run, that is, in the same play of the game. One of the sources of the conflict is agent's choice of action, i.e., if agent must choose between two actions a, b, then the event e_a corresponding to to the choice of a and the event e_b corresponding to b are in conflict. Moreover, if $e_a \rightarrow^* e'$ and $e_b \rightarrow^* e''$ (event e' is in the causal future of e_a and event e'' is in the causal future of e_b) then also the events e', e'' are in conflict. Another source of conflict are executions of joint action, which is explained later on.

Each two events can be either causally dependent, or in conflict, or unrelated by $\rightarrow^*$ and #. Obviously, these relations are disjoint as stated in item 5 of Def. 2.1. It is intersting that any two unrelated events, say e, e', are concurrent; we denote this as $e \| e'$. The *concurrency relation* relation is defined as $\| \overset{df}{=} (E \times E) \setminus (\rightarrow^* \cup (\rightarrow^*)^{-1} \cup \#)$.

We do not put any conditions on the labeling of events. Therefore, one can easily model non-determinism of actions. This is especially important as we need to model failure of joint actions.

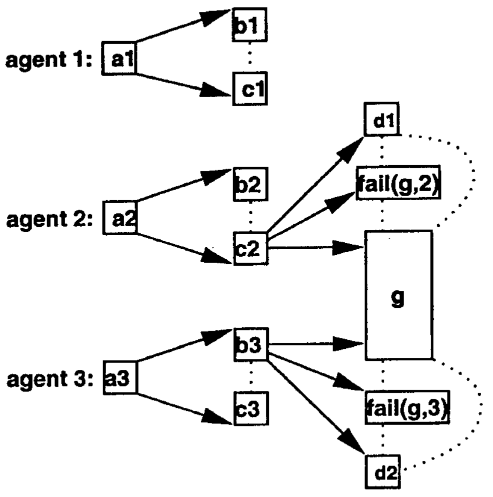
Figure 1 shows a local (lpes) and global representation of three player (agent) game, whereas Figure 2 gives some of the possible runs of the lpes.

Agent 1 is independent on the other agents, whereas agents 2 and 3 can synchronize by executing the joint action g. In case agent 2 schedules for execution action $d1$, then agent 3 will fail executing action g. Similarly, if agent 3 schedules for execution action $d2$, then agent 2 will fail executing action g.

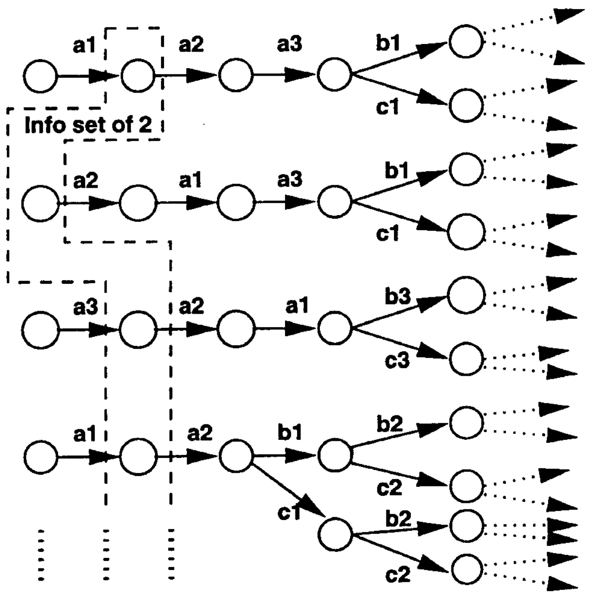
The local representation is given by an lpes, where the boxes represent events, the arrows the immediate causality relations, and the dotted lines the conflict relation. The conflict relation that follows from the conflict preservation condition is not marked in Fig 1. Labeling of events is given by letters put into the boxes.

The global representation is an extensive form of the above game. The circles represent global states of the game, and the moves of players are consecutive. Since there is a lot of concurrent actions here, all possible orders of moves must be present in global representation. Moreover, in order to define knowledge of a player (i.e. information sets in game-theoretic terms) the whole structure of global representation must be taken into account. This means that if we suppose a player to reason about knowledge, the player needs to know the whole global representation of the game.

It is a crucial point here to notice that in the case of local representation a different kind of knowledge can be defined, i.e., "causal knowledge." So that for player 1, his knowledge is only about himself. In fact he does not need to know about the other players. Moreover, he cannot have knowledge about others unless he has interacted with another player. Hence, in order to reason about knowledge, the player does not need to know the whole structure of the game, and even more the game structure does not need to be common knowledge of the players as it is assumed by Aumann and Halpern et al. So that in our framework of "causal knowledge" it is possible (and reasonable for large open systems) to



Local representation



Global representation

Figure 1: Example of the local and global representation of the same distributed game

allow the situation where initially players know almost nothing. They must acquire knowledge about the game played and the current play from performing actions, perception, and communication with other players.

For the purpose of our paper it is sufficient to assume that the set E is finite. Therefore, we can define the set of maximal (relatively to $\rightarrow$) events, called E^{term}. Let N be a finite number of agents and $E = E_1 \cup \ldots \cup E_N$, where E_i is a set of events agent i participates in, for $1 \leq i \leq N$. We assume that $(E_i \times E_i) \cap || = \emptyset$, which corresponds to the fact that the events of one agent cannot be concurrent. This, in fact, means that they are either causally related or in conflict. Similarly, $A = A_1 \cup \ldots \cup A_N$, where $A_i = \bigcup_{e \in E_i} l(e)$ for each i is the set of actions of agent i.

An event is said to be joint if at least two agents participate in that event. So that usually the sets $E_1, ..., E_N$ do not need to be mutually disjoint. For each event $e \in E$ let $agent(e) = \{i \in N \mid e \in E_i\}$ be the set of agents who participate in e. Since the events may be joint, also the actions that are associated to joint events are called *joint actions*. Similarly, for each action $a \in A$, we define $agent(a) = \{i \in N \mid a \in A_i\}$. The meaning of the joint action a is that in order to execute action a successfully, all the agents $agent(a)$ must participate in the execution. A joint action a can be executed if for all the agents needed for execution, the action a is enabled and selected for execution, i.e., intuitively all the agents "can" and "do want" to participate in the execution of this action. Let us note that the notion of joint action is closely related to the notion of simultaneous move of Kaneko & Dubey [6], however their approach is based on global states and global time. If one of the agents cannot or does not want to participate in the execution, then the attempt to execute the action a fails. For this reason we assume that for any joint action a, and any agent $i \in agents(a)$ there is a local action $fail(a, i) \in A_i$ that corresponds to agent i's failure to execute joint action a. It is natural to assume that if all the agents needed for an execution of a joint action schedule this action for execution, then the action will be executed. This is equivalent to say that any conflict-free behavior of an lpes in which all the agents i execute $fail(a, i)$ with $i \in agent(a)$, is forbidden. Maximal conflict-free substructures of lpes satisfying the above condition are called **runs**. This notion of run corresponds to the game-theoretical notion of **play**.

We define payoff function of agent i as, $H_i : E_i \cap E^{term} \longrightarrow Reals$

Definition 2.2 *A* **distributed game frame** *is the ordered triple* $F = (\mathcal{R}, (E_i)_{i \in N}, (H_i)_{i \in N})$, *where* $\mathcal{R}$ *is an lpes,* E_i *is the set of events of agent i, and* H_i *is a payoff function of agent i.*

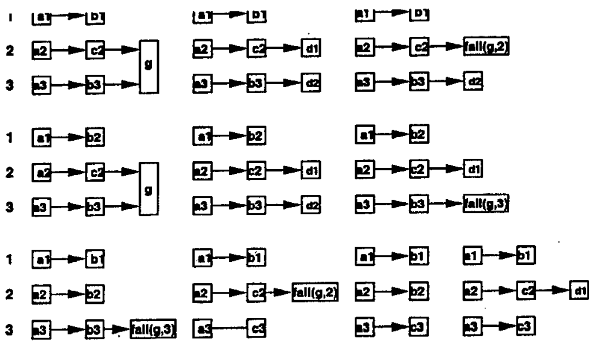


Figure 2: Runs of the event structure of the former example

In order to complete the definition of a distributed game we need to define information sets for each agent i. In the classical definition, the information sets of agent i are defined by a partition of his decision nodes. In our approach they are defined by a partition of the set E_i, i.e., the events in which agent i does participate and at which he also takes decisions concerning action choice. However, our idea of introducing causal knowledge relies on acquiring knowledge from perception and from communication with other agents. For this reason the information sets determined by agent's knowledge cannot be arbitrary. Agents' information sets must correspond to the cumulative and aggregated knowledge that is acquired in agent's history. Since agents acquire knowledge by communicating with other agents, in order to understand each other the agent must speak in common language with common semantics. Introducing such language and completing the definition of game is the aim of Section 3. In order to illustrate all the newly introduced notions, we present the game of Puzzles as the running example.

2.1 Puzzle game

Consider a network of computer servers and mobile software agents that can move from one server to another in order to look for resources (puzzle pieces) that are scattered randomly on the servers. To enhance cooperations between the agents, the entry to some servers is forbidden for some agents. Thus, if a resource the agent needs is on a forbidden server, the agent needs a help from other agents. On the other hand some agents may compete for the same resource. It is assumed that the cost of performing any action is one unit subtracted from agent's personal ac-

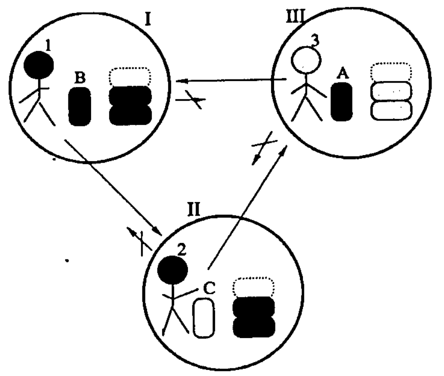


Figure 3: Example of Puzzle game

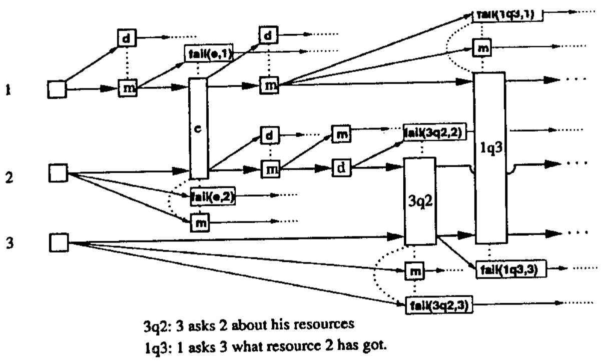


Figure 4: A part of the lpes for the example of Puzzle game

count. So, each agent can execute a finite number of actions limited by the introductory balance of his personal account. Each agent wants to get his resource with minimal effort (cost). Agents can execute the following types of actions: take (drop) a resource from (to) a server, get (give) a resource from (to) another agent, exchange a resource with another agent, move from one server to another, or end his activity switching to the 'off'–state. Switching to the 'off'–state is obligatory for an agent if his account is equal zero. Giving and exchanging resources is possible only for agents residing at the same server. Additionally any two agents can communicate (talk) to each other provided they are at the same server. Intuitively, Puzzle can be viewed as an example of a multi-agent system as well as of a game.

Let us notice that an exchange of resources r, r' between two agents i, j (which can be thought of as composed of two actions: a = (agent i gives r to agent j), and a' = (agent j gives r' to agent i) executed in either sequential order) is treated here as an atomic action for the following reason. After action a has been executed, there is no way to make self interested agent j execute action a'. Hence, the exchange cannot be modeled as a simple composition of two actions.

Initially, in the example of Fig. 3., each agent is supposed to know only his resources, his position, and which servers he can enter. So that initially he knows nothing about the other agents. In order to realize his goal player 1 needs A, player 2 needs B, whereas player 3 needs C. The lpes representing the behavior of the example of the Puzzle game is shown in Fig. 4. One run of the lpes is distinguished with the thicker lines. This run will be used as the running example of the next section.

3 Causal Knowledge

We assume that knowledge of any agent i can be stored in the variables $v_1^i, \ldots, v_K^i$ ranging over the sets $W_1, \ldots, W_K$. The sets W_k can be of any sort including: char, integers, reals, etc. This variable collection defines agent's knowledge frame. One can think about it as a relational database. It is important to note that the range of variables v_m^j and v_m^i is the same for any $i, j \in N$ and $m \in K$. The profile of current values of agent's variables (database) is considered as the current state of the agent mental state. So that all possible agent's mental states are collected in the set $M = \prod_{k \in K} W_k$. That is, for example $m_i = (w_1, w_2, \ldots, w_k)$ corresponds to agent i's mental state where $v_1^i = w_1, \ldots, v_k^i = w_k$.

Example 3.1 *For our game of Puzzle a simple database of each agent can be defined as a* 3×3 *matrix* $[v_{i,j}]$, *where rows store information about agents* 1, 2 *and* 3. *The information concerns the resources, the location of an agent, and the number of moves already performed by the agent. The star (*) denotes the lack of information.*

The variables $(v_1^i, \ldots, v_k^i)$ are updated or revised by agent i's perception and knowledge acquired from other agents via explicit communication.

Now we are going to define agent's perception. Let $P = \{p_1, p_2, \ldots, p_l\}$ be a set of common observable (propositions of Propositional Calculus) of all the agents N. That is, perception mechanism is of the same kind for all the agents. The propositions are evaluated at events in the following way: at e any agent $i \in agent(e)$ can evaluate any $p \in P$ and come up with the conclusion whether p is true or false or unknown.

Example 3.2 *For our game we can define the following propositions. For each* $i \in \{1, 2, 3\}$, $X \in \{I, II, III\}$, *and* $Y \in \{A, B, C\}$, *the proposition*

p_{iXY} is defined. The proposition is true if agent i is at server X and has resource Y. Agent i can evaluate proposition p_{jXY} at event e as true or false iff $i = j$ or the event e is a joint event of agents i and j. Otherwise the value of p_{jXY} is unknown.

Let $O = \{t, *, f\}^l$ (l is the number of propositions) be the set of all possible strings of length l over the set $\{t, *, f\}$ coding all the valuations of the propositions. For example, for $l = 4$, $o = ttf * t \in O$ corresponds to agent's observation that propositions p_1, p_2 and p_5 are true, proposition p_3 is false whereas the value of proposition p_4 is unknown.

Definition 3.3 *Agent i's perception is defined as a function $\Pi_i : E_i \to O$.*

An intuitive meaning of $\Pi_i(e) = ttf * t$ is that at event e agent i perceives the propositions in the way described above.

Example 3.4 *During the perception of agent 1 and 2 at the event labeled e corresponding to the exchange of resources B and C by agents 1 and 2 at server II, the agents perceive that the following propositions hold: p_{1IIC}, and p_{2IIB}. This means that after the exchange, agent 1 has got C and is at server II, whereas agent 2 has got B and is at server II, as well. As to the propositions p_{3IA} and p_{3IIIA}, the agents 1, 2 can not evaluate whether they are true or false. However they can deduce for example that p_{3IIB} and p_{3IIC} are false. This kind of deduction may be encoded in the memorizing or reasoning mechanism.*

Definition 3.5 *Common memorizing mechanism is defined as a function $Mem : M \times O \longrightarrow M$.*

An intuitive meaning of $Mem(m, o) = m'$ is that mental state m is updated to m' after observation o has been made by an agent.

Example 3.6 *After executing the event corresponding to the exchange of B for C with agent 2 at server II, the database of agent 1 is updated by the information about agent 2: at server II, resource B (see Fig. 4). Let us notice that with each observation of event the number of steps already executed by the agents participating in the event is updated in their data bases.*

Definition 3.7 *Let $Q = \{*, ?\}^K$ be the set of* queries. *The interpretation of query $q = * * *?*?$ is:* **tell me the value of variable v_4 and v_6 in your memory (database).** *The set of all possible answers to the queries Q is defined as the set $Val(Q) \overset{df}{=} \prod_{k \in K}(W_k \cup \{*\})$.*

Let the query $q = q_1 \ldots q_K$ be directed to agent i, whose memory state is m. The answer of agent i is equal to $Val(q, i) = x_1 \ldots x_K$, where $x_i = m_i$, for $q_i = ?$, and $x_i = q_i$ for $q_i = *$. For example the answer of agent i to query $* * *?*?$ can be $* * *2 * 9$.

Definition 3.8 Common reasoning mechanism *is defined as a function:*

$$\Xi : M \times Val(Q) \longrightarrow M$$

An intuitive meaning of $\Xi(m, x_1 \ldots x_K) = m'$ is that mental state m is updated to m' after receiving answer $x_1 \ldots x_K$ to a query sent by an agent.

Definition 3.9 *Agents' common* **ontology** *is defined as $\Lambda = (O, M, Mem, \Xi)$.*

It is natural to assume that any agent stores in his memory the information about the actions he may execute currently. This assumption will allow us to define coherently agent's information sets. Since the information sets represent agent's knowledge, the availability of actions must be stored in the knowledge.

Definition 3.10 *Game with local interactions, explicit communication, and causal knowledge* (**distributed game**, *for short) is defined as $(F, (\Pi_i)_{i \in N}, \Lambda)$, where F is a distributed game frame, Π_i is the perception of player i, and Λ is a common ontology.*

When an agent gets information from another agent, then he puts it to his data base in two cases. Either, he has not had any information on this subject or he is able to infer that the information he has just received is more recent than his own. Let us notice that the reasoning mechanism as well as the memorizing mechanism is common for all the agents. The reason for this is mainly technical. If these mechanisms had not been common, then in order to have common semantics (ontology), translations from one agent's language to other would have been required.
In the set A of actions we identify a subset of *communication actions*. For each query $q \in Q$ we have a joint action iqj of agents i and j with the intended meaning of agent i sending query q to agent j. If agent j does agree to answer the query then the action iqj is successfully executed, otherwise it fails.

Example 3.11 *In the Figure 5 we indicate a possible run of the lpes representing the behavior of our game. We assume that each agent has performed some initial local event. All events are first drawn as local ones. Joint events are obtained by grouping the corresponding local events. Each event, except for the initial*

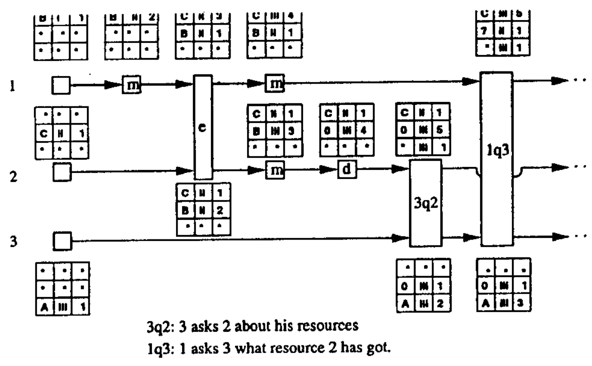

1
2
3
e
1q3
3q2
3q2: 3 asks 2 about his resources
1q3: 1 asks 3 what resource 2 has got.

Figure 5: One run indicated in the lpes for Puzzle example

ones, is labeled by the first letter of the name of the action, which was executed at this event. Moreover, each event is equipped with the 3x3 table storing the agent's knowledge. Let us note that the figure shows an updating problem. The agents 1 *and* 3 *execute the joint event labeled by action* $1q3$, *where q is a query: "What resource has agent* 2 *got ?" Then, agent* 1 *cannot update his data base since he does not know, who has got the most recent information about agent* 2. *This is marked by the question mark (?) The way of solving this kind of problems is by encoding an extra information, which is sufficient for deciding who has got the latest information about other agent. One of the possible solutions is so called secondary information defined for gossip automata [11].*

3.1 Information sets

It is supposed that the agents use a fixed common ontology when they play a distributed game. So that having specified perception for each agent, any agent changes in the course of play his mental state according to perception, memorizing mechanism, communications with other agents and reasoning mechanism. Hence, any $m \in M$ uniquely determines an information set of agent i. It is possible that this set is empty, if the mental state can not to be reached by agent i.

Thus, the information sets may be identified with the elements of M.

Let $E_i(m)$ be the set of all events, where the mental state of agent i is m. It is clear that the collection of $E_i(m)$ for $m \in M$ is a partition of E_i.

The assumption that any agent stores the information about current availability of actions, guarantee the coherence of the notion of information sets, that is, any agent must know what actions he may execute currently.

One may ask how distributed games relate to the classical games in extensive form. As far as we consider the definition of game frame, it is a matter of taste. That is, both approaches, the classical one and ours model the same situations, i.e., the same real games. In this sense they are equivalent, i.e., one can be transformed into the other one. However by introducing causal knowledge, we narrow the class of games under consideration. The gain is that we can explicitly consider agent's knowledge, and we can model and simulate plays of large and open games.

4 Analogs of Kuhn's theorems

As to the first theorem of Kuhn (see [7]) about existence of Nash equilibrium point in games of perfect information, the notion of perfect information in the classical definition cannot be implemented in our framework of distributed game unless we assume that there is no concurrency in the system. This means that all the agents participate in all the events.

The maximal knowledge an agent can have is when all agents exchange all their knowledge during each synchronization, and have capability to store all the acquired knowledge. This maximal causal knowledge does not correspond to perfect information in games, because an agent can not have a knowledge about another agent unless there is a causal relation between them.

Now we are going to show that the relations between mixed and behavioral strategies are the same in both the distributed and the classical case.

For any subset $A' \subseteq A$, let $\Delta A'$ denote the set of all probability distributions over set A'. Let $A_i(m_i)$ be the set of actions available for agent i to take at his information set m_i. However, the sets $A_i(m_i)$ do not include the failures of joint actions, that is, $fail(a, i)$ is not an action for itself and can not be executed by an agent. It is a result of unsuccessful execution of action a by agent i. So that $fail(a, i)$ is a means for representing indeterminacy (success or failure) corresponding to execution of action a.

Let us note that according to the assumption made below Definition 3.9, the set $A_i(m_i)$ is uniquely determined. That is, for all $e \in E_i(m_i)$, the set $A_i(m_i)$ is the set of actions available for agent i to take at e.

We define pure local strategies (s^i) as functions of the form:

$$s^i : M_i \longrightarrow A_i$$

such that $s^i(m_i) \in A_i(m_i)$, and S^i denote the set of pure local strategies of agent i.

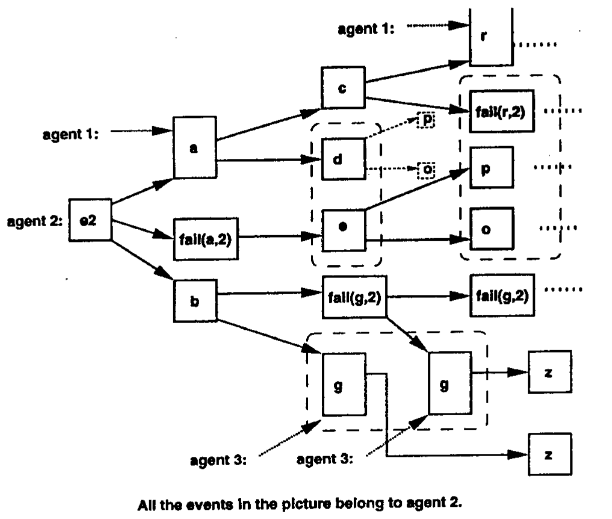


Figure 6: Information sets of agent 2 in distributed framework.

Let $S = S^1 \times \ldots \times S^n$ be the set of n-tuples (or profiles) of pure local strategies.
Let us recall that we exclude the situation in a play (run) where all agents $agents(a)$ want to execute the action a and they fail. In this case the action a is executed successfully. As a result of this assumption, we have the following lemma.

Lemma 4.1 *Each profile $s = (s^1, s^2, \ldots, s^n) \in S$ of pure strategies uniquely determines the subset of events of E, denoted $E(s)$, that are reachable under s. Moreover, $E(s)$ contains exactly one terminal node for each agent i.*

For n-tuple $s = (s^1, s^2, \ldots, s^n) \in S$ of pure strategies, the payoff $h^i(s)$ to agent i is defined by

$$h^i(s) = H_i(e_i)$$

such that $e_i \in E(s)$
The set of *mixed local strategies* X^i of agent i is defined as $X^i = \Delta(S^i)$, where S^i is the set of pure local strategies of agent i. This means that agent i chooses each pure local strategy with probability $x^i(s^i)$. Let $X = X^1 \times \ldots \times X^n$ be the set of n-tuples of mixed local strategies.
For n-tuple $x = (x^1, x^2, \ldots, x^n) \in X$ of mixed strategies, the payoff $h_i(x)$ to agent i is defined by

$$h_i(x) = \sum_{s \in S} x(s) h_i(s)$$

where $x(s) = \Pi_{j \in N} x^j(s^j)$ is the probability, under x, that the profile s has been selected by the agents.

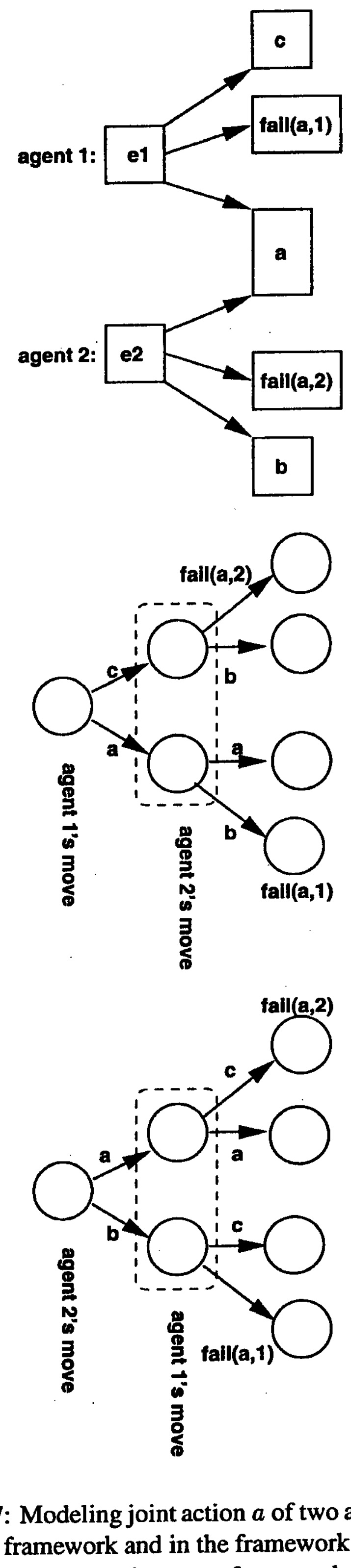


Figure 7: Modeling joint action a of two agents in distributed framework and in the framework of extensive games. In the extensive game framework two possible orders of moves must be considered.

We define behavioral strategies b_i of agent i (corresponding to the behavioral strategies in the classical definition) as functions of the form:

$$b_i : M_i \longrightarrow \Delta A,$$

such that $b_i(m_i) \in \Delta A_i(m_i)$.
We write $b_i(m_i; a)$ for the probability of the choice of action $a \in A_i(m_i)$ by agent i at m_i . The set of all behavioral local strategies of agent i is denoted by B_i. As in the standard approach the set B_i of behavioral strategies of agent i can be identified with a subset of the set X^i of mixed strategies. Formally, the mixed local strategy x^i corresponding to the behavioral strategy $b_i \in B_i$ is defined by $x^i = (x^i(s^i))_{s^i \in S^i}$, where

$$x^i(s^i) = \prod_{m_i \in M_i} b_i(m_i; s^i(m_i))$$

Hence, for any profile of behavioral local strategies $b = (b^1, b^2, \ldots, b^n)$ we define

$$h_i(b) \overset{df}{=} h_i(x)$$

Let us note that in the definition of mixed as well as behavioral strategies of agent i involves only the structure of information sets and the agent's actions. The same is true for the standard definition. In the case of distributed games the information sets are defined as subsets of the set of agent i's events, whereas in the standard definition the information sets of agent i are subsets of the set of his decision nodes. Clearly, the events of agent i are at the same time his decision nodes, that is, the places where agent i chooses an action to execute.
However, the structure of information sets in the case of distributed games is simpler that in the classical case. The reason is that from the point of view of a single agent the only source of indeterminacy (in the distributed case) is the execution of his joint actions, see Fig. 6, whereas in the classical case the next moves of other agents. As it it can be seen in the Fig. 7, successful execution of joint action and failure can be classically modeled as consecutive moves of agents to whom this action belongs. Hence, any distributed game can be represented (modulo orders of players' moves) as a classical extensive game, where the structure of the corresponding information sets are in fact the same as in distributed case. The information sets structure in the classical representation is extended by the information sets that correspond to the joint actions as shown in Fig. 7. We are going to the conclusion that the definitions of mixed and behavioral strategies in both the classical and the distributed case are actually the same. This is because any distributed game can be represented as a classical extensive game with actually the same structure of information sets. Thus also the notion of perfect recall is the same in both cases. Hence, there is equivalence between the behavioral and mixed strategies in distributed games, the same as it is in the classical case proved by Kuhn [7]. So that for any mixed strategy of an agent there is a corresponding behavioral strategy of the agent that gives him the same payoffs.

Theorem 4.2 *Let Γ be a distributed game in which agent i has a perfect recall. Then, for each mixed strategy from $x^i \in X^i$ there is a corresponding behavior strategy b_i that is equivalent to x^i.*

5 Conclusions

A new definition of game in extensive form based on local agents' interactions and casual knowledge has been presented. The main difference with the classical notions of knowledge by Aumann, and Halpern et al. relies on restricting information sets in such a way that players are not supposed to know a priori the whole structure of the game, so that their knowledge about the game played and the play is acquired in a causal way. Our definition can be seen as a revision of the classical definition rather than an extension.
This approach has several consequences on the theoretical and practical ground. The first theorem of Kuhn does not hold in our approach since the condition of perfect information cannot be imposed at every distributed game. However, the second theorem of Kuhn still holds.
As far as practical advantages of our approach are concerned, we should mention a very compact representation of games. This feature allows for easier and more efficient analysis and verification of game plays. As to the the notion of knowledge we have introduced, it seems that this is a reasonable notion that can be used for modeling agents in large, distributed, and open environments.

References

[1] S. Ambroszkiewicz, O. Matyja, and W. Penczek. Team Formation by Self-Interested Mobile Agents. In *Proc. 4-th Australian DAI-Workshop,* Brisbane, Australia, July 13, 1998. Published in Springer LNAI 1544. http://www.ipipan.waw.pl/mas/

[2] R. Aumann. Agreeing to Disagree. *Annals of Statistics,* 1976, Vol 4, No. 6, pp. 1236-1239.

[3] D. Bell, and J. Grimson. *Distributed Database Systems.* Addison-Wesley Pub. Co., 1992.

[4] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about knowledge*, MIT Press, 1995.

[5] S.Hart, Games in Extensive and Strategic Form, in *Handbook of Game Theory,* Edited by R.J.Auman ans S. Hart, Elsevier Science Publishers B.V., 1992.

[6] M. Kaneko, and P. Dubay. Information Patterns and Nash equilibria in extensive games: Part I. *Mathematical Social Sciences 8* (1984), 111-139. Part II 10 (1985), 247-262.

[7] H. W. Kuhn (Ed.). Classics in Game Theory. Princeton University Press 1997.

[8] R.E. Ladner and J.H. Reif. The logic of distributed protocols, Proc. of Tark 1986, pp. 207-221.

[9] K. Lodaya, K. Parikh, R. Ramanujam, P.S. Thiagarajan, A logical study of distributed transition systems, Information and Computation, vol. 19, (1), 1995, 91–118.

[10] K. Lodaya, R. Ramanujam, P.S. Thiagarajan, "Temporal logic for communicating sequential agents: I", Int. J. Found. Comp. Sci., vol. 3(2), 1992, pp. 117–159.

[11] M. Mukund, and M. Sohoni. Keeping track of the latest gossip: Bounded time-stamps suffice, *FST&TCS'93, LNCS* **761**, 1993, 388-199.

[12] W. Penczek, A temporal logic for event structures, *Fundamenta Informaticae* XI, pp. 297–326, 1988.

[13] W. Penczek, A Temporal Logic for the Local Specification of Concurrent Systems. *Information Processing IFIP-89*, pp. 857– 862, 1989.

[14] W. Penczek. Model checking for a Subclass of Event Structures, Proc. of TACAS'97, LNCS 1217, Springer-Verlag, pp. 145–164, 1997.

[15] W. Penczek. A temporal logic of causal knowledge, Proc. of WoLLic'98, pp. 178–187, 1998.

[16] R. Ramanujam. Local knowledge assertions in a changing world. In *Proc. of the Sixth Conference TARK 1996, Theoretical Aspects of Rationality and Knowledge,* Y. Shoham editor, pages 1-14, Morgan-Kaufmann, 1996.

[17] A. S. Rao and M. P. Georgeff. Modelling rational agents within a BDI–architecture. In R. Fikes and E. Sandewall, editors, *Proc. of the 2rd International Conference on Principles of Knowledge Representation and Reasoning (KR'91),* pp. 473–484, Cambridge, Mass., April 1991, Morgan Kaufmann.

[18] Winskel, G., An Introduction to Event Structures, LNCS 354, Springer - Verlag, pp. 364-397, 1989.

[19] W. Zielonka. Notes on finite asynchronous automata. *RAIRO-Inf. Theor. et Appli.,* vol 21, pp. 99–139, 1987.

A DECENTRALIZED MANAGEMENT OF THE PRODUCTION RESOURCES FLOW BASED ON MULTI-AGENT MODEL

Viktor V. Emelyanov

BMSTU, 107005, Moscow, st. 2-nd Baumanskaya, h.5, RK-9 *evv@rk9.bmstu.ru*

Abstract

The aim of the paper is the development of the new approach to designing a decentralized management system which performs the exchanges of industrial resources based on the model of collective behavior of intelligent agents. The results of the construction of a multi-agent architecture for the management system are represented. The abstract intelligent agents underlay this model. Theirs interaction and activities determine a required intelligent behavior of the whole multi-agent system. These agents realize the creation of some required types of resources and interchange by them for reaching of the global purpose of the production system. For the research of interacting agents the intelligent simulation tools were used. The work is supported by Russian Fund of Basic Research (grant 99-01-01136).

Keywords: *CIM, intelligent agent, multi-agent system, management, simulation.*

1. Introduction

The automated management system of the computer aided integrated manufacturing (CIM) consists of a set of local management systems (LMS), functioning in a computer network of the firm (or its part). All the production operations are realized under the control of LMS. Their realization requires performing the following stages:

- Stage 1. Definition of the operation which is necessary to be executed.
- Stage 2. Supply of this operation by required industrial resources:
- Stage 2.1. A choice of the CIM elements as resources of operation.
- Stage 2.2. The management of auxiliary operations, which are realized by moving the selected resources from a state to the required one.
- Stage 3. Realization and successful completion of the operation.

For reaching general purposes faced by CIM, the coordination and cooperation of operations performed by separate LMS should be ensured. The LMS are distributed in a computer network The process of the LMS coordination and cooperation can be carried out various methods, which differ one form another first of all by the level of centralization. The centralized systems have some uniform control device, but in decentralized systems the control is realized at the cooperation of LMS [Emelyanov 1990, Gornev *et al.*, 1990].

Here CIM represents a constantly varying system. The modifications are dynamical and are frequently difficult to predict. It means that the CIM is evolving as a system. The complicated system which has arisen by an evolutionary way, can not cope by the uniform centralized management system [Varshavskiy *et al.*, 1984; Emelyanov 1997]. This will be the CIM management system that should be created as a system with a pre-defined level of decentralizing. On the other hand, the decentralized mode of the work requires higher flexibility and adaptability by everyone by separate LMS. This requirement can be carried out at organization LMS as intelligent [Lefrancois *et al.*, 1994; Parunak 1993].

In the present work the intelligent decentralized CIM management system is considered as the multi-agent system. Here the LMS are considered as independent agents, which have a pre-defined rights specifying their level of intelligence and the possibility to make decisions, using for this purpose an available information and knowledge. Has a place a collective behavior of the intelligent agents (IA), when each IA communicates with another ones to obtain the necessary resources for performing industrial operations. Horizontal connections have priority over vertical ones in processes; IA are separated.

The simulation methods are suggested for researching collective behavior of intellectual agents. Such approach to building simulation model of decentralized control system is based on intellectual simulation, using artificial intelligence in a hierarchy of interacting systems and analysis of mutually dependent processes. The main subjects of the model are abstract agents in the organization - their activities and interaction determine the required collective intellectual behavior [Artificial 1996; Tarasov 1996].

RAO intellectual simulation medium was chosen as a tool for simulation [Artiba *et al.*, 1998; Emelyanov et al.,1997].

2. The Multi-Agent Approach to Decentralized CIM Management Systems

Decentralizing of management is related to the fact that all LMS make decisions under uncertainty when the information about other LMS is incomplete and ambiguous. Here the LMS ensures a real-time control of all elements in the cell and searches for additional resources located in other cells. For this purpose LMS should made decisions about using its own resources and the resources it needs. The missing resources are at the disposal of others LMS and can be transferred for realizing the industrial operations above. The local management systems in CIM is linked to all other LMS. This LMS also should select the best plan among the set of all possible plans. Therefore, the set of LMS should exchange the information intensively for organizing a purposeful CIM work in the whole. Thus, separate LMS have a complete information about their own resources and purposes, but they have a few information about purposes and effectiveness of the whole system.

Therefore, among the most significant functions of decentralized management system, we point out the following:

- Connection with a management system of a higher level. In spite of the fact that we create our system as a decentralized system, the links with the management systems of higher hierarchy level can not be completely lost. Our task is the elimination of a governing role of a higher level system during the real-time management. The role of a management system of a higher level should be reduced to the definition of the production strategy and the plan distribution at the beginning of changes between LMS.
- Monitoring of resources and services. The process of providing cells with the resources has two aspects: A - definition of necessary resources and as a consequence, ordering of missing resources; B - mechanism of searching for and delivery of necessary resources. When LMS orders required resources, it should not care of their location and state.
- Detection and resolution of conflicts. During the LMS interactions, various conflict or disputable situations may occur (for example, concerning the order of using resources). To detect and face such problems, it is necessary to introduce into the system some additional functions.
- The modification of a system. Because CIM is a constantly varying object, it is necessary to add some new modules. The structure of available systems can vary, as well as the topology of their connections in a system.
- Functions of distributed database for the accumulation of both knowledge and data concerning the state of the system.

Thus, every LMS should work independently with analyzing its own state and making decisions; so it needs to be autonomous and active. Everyone LMS should work in an optimal way to attain the goals, such that the global CIM purpose would be achieved (the more efficiently, if possible). To prevent conflicts, which are unavoidable in the autonomous work case, LMS should perceive the external world state and react operatively to its modification; it also should be connected with other LMS and accepted by them. The autonomous mode of functioning also needs learning capacities. Therefore, the LMS may be viewed as an intelligent agent possessing such properties as autonomy, reactivity, activity, social behavior, life in the network, flexibility, adaptability etc. [Braspenning 1997; Jennings 1995; Zweben 1997]. The necessity of representing these properties in LMS can be shown via the analysis of their operations in CIM [Fleury *et al.*, 1984].

The use of a multi-agent architecture allows decisions to be taken in a decentralized way. In artificial intelligence, this approach appears to be well suited to complex problems, especially those with a great

number of interactions between components, and for which classical incremental methods cannot provide good results. The multi-agent method allows one to solve subproblems locally with an agent, and to propose a global solution as a result of interactions between the different agents [Kouiss 1997].
For the production management of a CIM, many decisions have to be taken to reach the production objectives (e.g. planning decisions, scheduling decisions, and control decisions). Indeed, these decisions must be periodically updated in order to take into account changes in the CIM (e.g. a machine breakdown, worker absences, etc.). The use of a multi-agent architecture allows one to share out all these decisions between several agents in a hierarchical manner. Each agent is in charge of specific decisions [Chandra *et al.*, 1991; Iffnecker *et al.*, 1991; Baptiste *et al.*, 1993; Kwok et al., 1993; Barbuceanu *et al.*, 1994; Tacquard *et al.*, 1994; Trentesaux *et al.*, 1995; Ouzrout, 1996]. Unfortunately this structure presents some disadvantages, due mainly to possible contradictory decisions of agents that can lead to a global lock of the system [Attoui *et al.*, 1995].
The obvious complementarity of the multi-agent and simulation approaches to management problem has led to a growing interest in these hybrid systems involving the cooperation of both approaches. Apart from the usual reasons for building a multi-agent architecture, some conditions were specially taken into account, such as modularity and ease of integration [Larry 1995].
Distributed heterogeneous systems are receiving great attention from researchers in all the computer-related fields [Velasco *et al.* 1995; Virdhagriswaran 1994].

3. Types of the Agents in the Decentralized Management System

Agents may be positioned in a system by using technological, functional, topological or organizational criteria. In our case, IA overlays CIM topology. Each IA is an entity able to modify its environment and, if necessary, to interact with other agents. To do it, the IA uses its knowledge base and inference mechanism. The goal of each IA is to simulate the decision that may be made by a human for a given CIM system. Here the IA uses heuristic methods and takes into account what products have already been manufactured, what resources (its own or borrowed) are available or ordered, what data depend on the decisions made by other IA, what due-dates are planned etc.
In the basic workflow model in the CIM system we shall consider the interaction between two agents, the Customer and the Performer. Both are the instances of same IA type, the resource USER agents. The USER agents function independently and pursue their own goals (performing prescribed basic operations), with using their knowledge (heuristic methods), data and criteria. So the USER agents are active IA because nothing initiates their operations from outside.
As a Customer, the USER agent sends requests to all the other USER agents requiring that they either get some resources from it or send some resources to it. Another USER agent, while receiving such a request and, after analyzing it, can either accept it for fulfilling (putting it into the queue of its basic operations and hence becoming Performer in the interaction between two IA) or reject it. After receiving a request IA sends a relevant message to other USER agents. As a result of such information exchange, the interaction is realized between two IA, the Customer and the Performer. Any USER agent may be at the same time both Customer and Performer in respect with the other USER agents. Here Customer and Performer are the active IA in the system - that means that they initiate their own operations and the operations of other IA.
For delivering requests and responses, a Channel agent (or agents) is used. Its purpose is to provide information exchange for all the other IA. The Channel agent delivers a request or a message to a given address (if there is any) or to the all IA in the system (if there is no specific address). So the Channel agent takes functions of a various packages (information, controlling, introducing) routing. Using the knowledge the Channel agent can optimize various retrieval operations and process the distributed information in a network.
For the storage of the resources and re-distributing them among the subsystems, some transportation and storage systems are necessary in the CIM. In multi-agent model, their LMS are represented as Service agents. The USER agents are able to negotiate with the Service agents about receiving, sending or moving resources. The Service agents receive requests for transporting, giving out or storing resources. These Service agents are passive - they don't initiate other IA's actions and don't seek them. The function of agent's interaction coordination, as well as the detection and solution of conflict situations is made by a Supervisor agent. Also his task

consists in the correction of the system while the new agents emerge. To connect agent's cyberspace with an exterior environment an Interface agent is introduced. He is the mediator between the various agents and operator (Fig.1).

Thus, a basic model of multi-agent system seen as a functional - structural unit of a decentralized management system is suggested, including the interactions between five basic types IA (User, Channel, Supervisor, Service, Interface), which have the connections of three types (Coordination, Controlling and Informing).

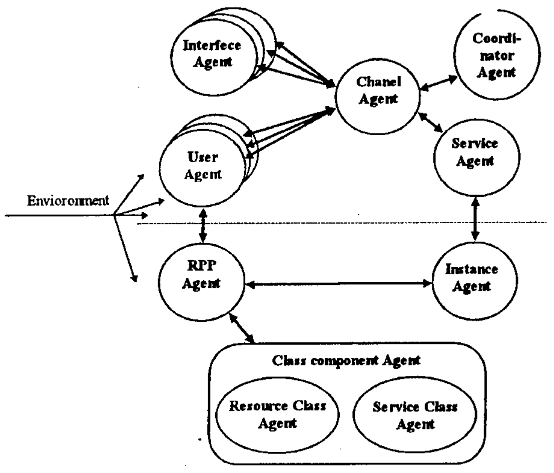


Fig. 1. Architecture of multi-agent distributed management system

4. Multi-agent Model of the Resources Exchange Process

We shall consider the following types of resources: finite activity products, if we deal with a producing cell of the CIM; services, if the deal with the transport cell or some other auxiliary cell; parts or free places for them, if the storage system is considered, etc. The resources have two basic parameters: the site and the state. To obtain or consider the opportunity of receipt of a required resource, the IA called the Customer should know the meaning of these parameters. However, in this case, when the given parameters constantly change and the set of resources (even identical) is very large, the accumulation of such huge information amounts becomes practically impossible.

Therefore it is necessary to create the flexible mechanism of initialization of resources exchange. Also it is necessary to have the mechanism of dynamic adjustment of parameters for the resources exchange process when the CIM system works.

First of all it is necessary to pass the knowledge about the resources structure and state from the user (Customer) to the resources themselves (or to the place of their birth). Furthermore, it is necessary to organize the resources interaction and finally to entrust supplying the resource components to the resources themselves. So we convert the resource into active objects, which can be naturally represented as IA. Thus, both the resource itself (material, information or resource - operation), and its virtual counterpart are present in the system.

To realize the multi-agent resources supply system, the following new agents types can be introduced (Fig.1):

•Resource provision process agent (RPP agent) - initiates and co-ordinates the process of resource maintenance for the User agent, playing the Customer role. It is generated by the User agent, which appears as the resource Customer.

•Class component agent (CC agent) is the agent containing the knowledge on the resource components and parameters. Because the system possesses multi-component resources and there can be some variants of combining these components, we obtain the resources class. Let us introduce the proper IA resource A, as well as the resources, owned by others IA. These resources are required (Fig.2) to supply the resource R. Two combinations of these resources {B, C} and {D, E} are possible, each of them allowing the construction of the resource R. Depending on a selected combination, some various instances of the resource class can be obtained.

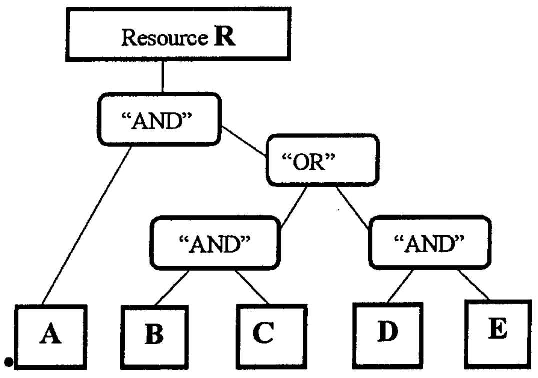


Fig.2. An example of a structure of the resource class components

Each agent belonging to the class of component agents "is linked" to a User agent or Service agent. Therefore it is possible to select two types of CC agents:

1. Resource Class agent (RC agent) - contains knowledge on a current state and technological

structure of a resource. This agent is linked to the User agent, which creates or stores a given resource.
2. Service Class agent (SC agent) - contains knowledge about a current state and technology of a service operation. This agent is linked to the User or Service agent, which implements the given service operation.
Instance agent – selects the components of a required resource. It is formed by the decomposition of the Class component agent through the digitalization of a resource parameters. The Instance agent is the direct initiator of those or other operations in a system. It addresses the call to make up the operation to the User or Service agents. The operations should be feasible by the reference moment. The Instance agent is supported by the Class agents during the previous phase of resource maintenance process.
The set of all class attributes may be represented in a natural way by a frame.
It is suggested to introduce the set of the frames, which will reflect a structure of a class. This set contains real values of the class parameters at various stages of work. The top level frame will be referred as the "Frame of resource class " (Fig3).

Prototype frame of resource class			
Slot	Slot name	Meaning	Description
1	Provider name		Name of User or Service agent
2	Type of resource		Name of resource type, which support its class
3	Resource parameters		Reference to frame of basic type of class resource
4	List of sets class component resources		Names of sets
5	Name of active set		Name of active set of components in class instance
6	Class accessible		YES of NOT

Fig.3. Prototype frame of resource class

The following frame contains the information about a component of the resources set. This frame is called "Frame of resource components set".
Each class component is a basic resource for any other class. Moreover, there can be a variety of such classes. They are unified by their membership to the same resource type, and distinguished by the name of User or Service agents which are providers of a given class.

5. Interaction of the Agents

The process of resources exchanging in multi-agent system generally can be founded on the workflow model. In this model the basic cycle links the Customer agent with the Performer agent. The cycle consists of four phases called workflow: "preparation - negotiation - performance - acceptance ". At the preparation stage the Customer addresses a demand (or an order for work realization) to the Performer. At the following stage the negotiation takes place directed at the development of mutually acceptable conditions of the order satisfaction.
Then the Performer performs the job and by ending it, reports to the Customer on its completion, and at the last phase the Customer accepts or rejects the job.
Here the engagement network appears that forms an organizational structure, and the feasible communication acts of the agents giving the communication the protocol, are determined by possible speech acts.
Applying to the model of resources exchange, the following cycle of operations is suggested. On the basis of its knowledge or its behavior model the User agent makes the decision about the initialization of some industrial activity. To do this work he needs some additional resources. In this case he will try order such absent resources. The ordering process, for a single resource, will consist of the following steps (Fig.4):
•User agent generates the RPP agent, which is responsible for the coordination of the resource supply process. To do this the RPP agent is provided with all the necessary knowledge, both about the resource, and concerning the mechanism of interaction with other agents.
Each of RC agents, after receiving the request, specifies its correspondence to his own resource. When the correspondence takes place, the class becomes active and produces the operations necessary for a concrete definition of all parameters of the basic resource. If these concrete parameters correspond to the specification, then the RC agent forms an answer to the RPP agent and generates the Instance agent, which, in its turn, will initiate and coordinate the User or Service agents operations to provide a required resource.
In the correspondence with the above-stated scheme of the agents interaction in the resource providing system the resources, each agent performs a se-

quence of pre-defined operations in a restricted area. All these operations are initiated and introduced on the basis of the agent internal knowledge, with the use of inference module and the mechanisms of interaction with other agents. The state diagram for the agents is represented in Fig. 5. The arrows designate possibilities of the agent transition from a state to another one. From this diagram the possible sequence of operations produced by the agents is obvious.

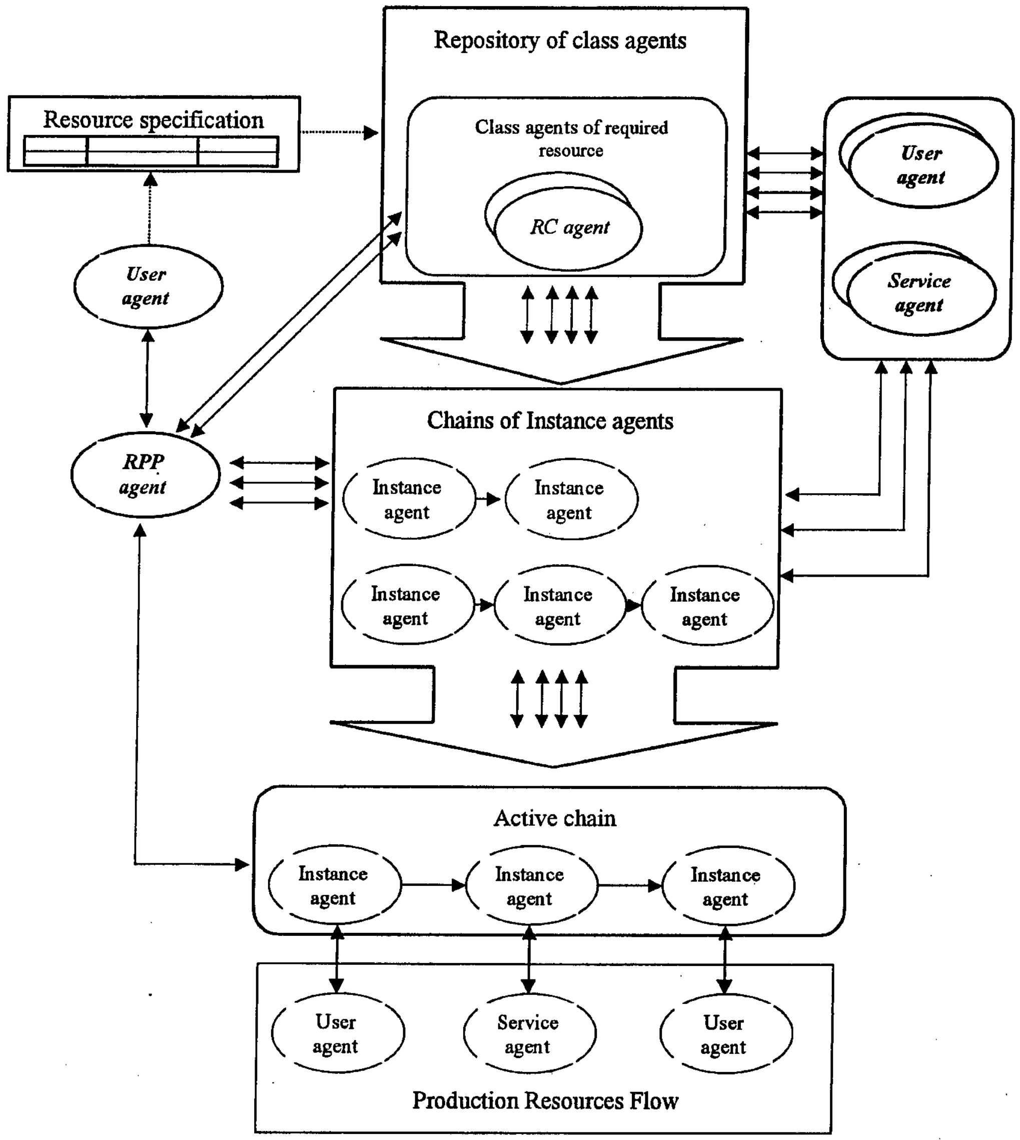


Fig.4. A multi-agent model of resources exchanging

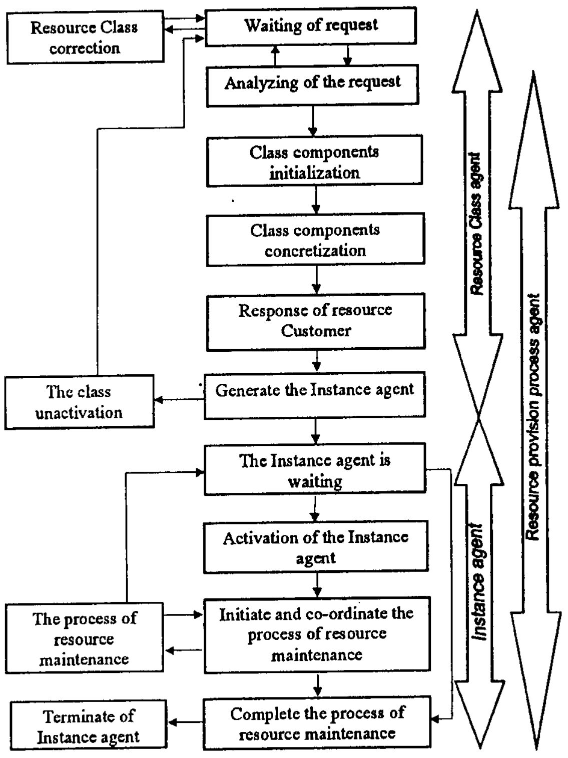

Resource Class correction
Waiting of request
Analyzing of the request
Class components initialization
Class components concretization
Response of resource Customer
The class unactivation
Generate the Instance agent
The Instance agent is waiting
Activation of the Instance agent
The process of resource maintenance
Initiate and co-ordinate the process of resource maintenance
Terminate of Instance agent
Complete the process of resource maintenance
Resource Class agent
Instance agent
Resource provision process agent

Fig.5. The state diagram for the multi-agent system

6. RAO Method (Resources - Actions - Operations)

At present, our approach has not been implemented on real CIM. The simulation model of multi-agent systems was designed to research its performance. We take a new method of knowledge representation and use - RAO (Resources-Actions-Operations), applied for the exposition and simulation of complicated discrete systems, construction of control systems, development of hybrid systems etc. [Artiba *et al.*, 1998, Emelyanov *et al.*, 1997].

Following the RAO-method, the model of a complicated discrete system is given by a dynamic production system. The database for such a production system contains the information about the system resources, and the knowledge base includes the set of operations fulfilled by these resources. The adaptation to a concrete simulated system consists in the formal representation of resources and operations on some language with introducing them into the knowledge and databases.

For the purposes of intelligent simulation, the knowledge in RAO language, using modified production rules, is written as follows:

IF (condition) THAT (event 1) TO WAIT Δt THAT (event 2).

The events 1 and 2 represent events of the beginning and termination of some operation having the temporal duration Δt. The operation time depends on the state of resources in a simulated system. The operation can be interrupted by other operations (mainly by irregular events).

The inference mechanism works with modified production rules which permit to construct a model of the process and to construct feasible and optimal solutions.

All main components of the discrete system that is its elements, process, rules of functioning are related in RAO-method to the following information objects: resources, actions and irregular events, operations. The complex system model is created from the above-mentioned elements by a set of resources and a set of operations. The process model is presented by a temporal sequence of actions and irregular events.

The simulation model may be created when the mechanism of events and irregular events simulation block are added to the dynamic production system. Besides the above-mentioned elements the simulation system contains a subsystem of collection of performance measures, which is intended for collecting results and their primary processing.

This RAO language due to its flexibility allows to describe in the framework of a uniform universal approach both decision made by separate IA, and the fruits of their interaction during the information exchange, taking into account the process dynamics. Besides, there is a possibility to describe in the same formalisms the whole production process in the CIM. It allows us to estimate the effectiveness of the control process.

7.RAO-model of Interaction Class Agents

We shall show some designs of simulation model of multi-agent management system and show of a little bit various elements of model in environment RAO.

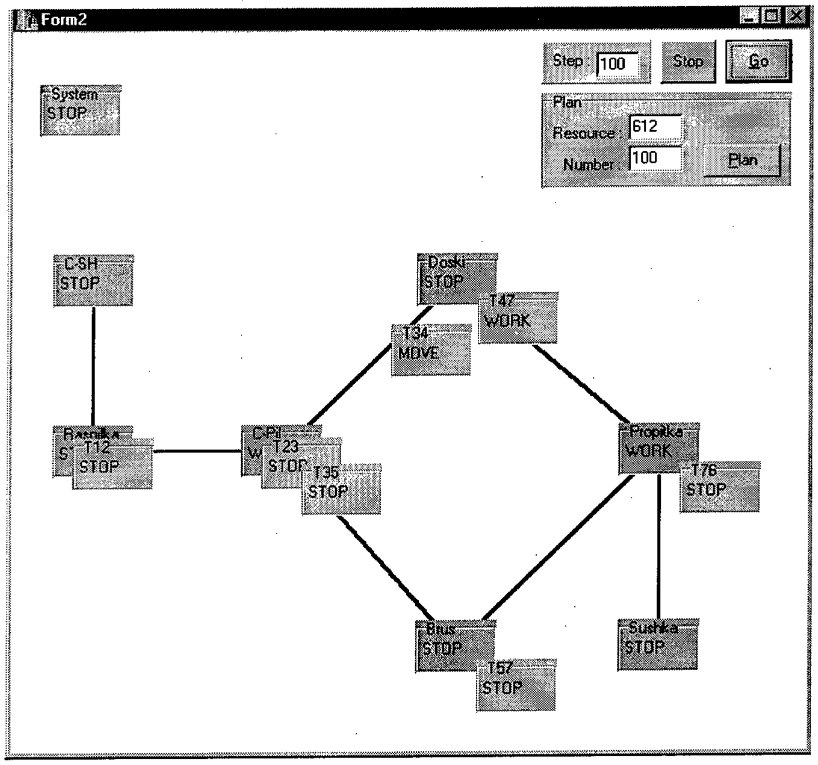
All IA, inquiries, messages, the tasks and etc. are represented by resources of RAO-model. Each resource should concern to which or type. The type of a resource determines structure of its attributes, their possible meanings, meaning by default. There are 8 resource types in the model: *Resource_Class, Service_Class, Instance, User, Service, System* and *Node*.

For example the description of *Resource_Class* as resource type, which is permanent (that is all time is in system) looks like:

```
$Resource_type Resource_Class: permanent
  $Parameters
     Number     : (A, B, C, D, E, F, Z)
     Owner      : (Ma, Mb, Mc, St)
     Graph_no   : integer
     State      : integer = -1
     UsC        : integer = 0
     Ct         :integer =0
$End
```

All actions, which are carried out in model by resources and over resources are described in object patterns of operations. Each pattern represents usual or modified production rule. All operations of one kind in model should concern to a certain pattern and differ by only used resources (relevant of operation) and parameters.
So the pattern of decomposition *Resource_Class* looks like:

```
$Pattern      R_ClassDecPr: rule trace
$Relevant_resources
        Tr: Resource_Class      Keep
        Sys:Sys1                Keep
$Body
Tr
        Choice from
        Tr.State = -1
first
Convert_rule
        State   set     -2
        { Selection of class for decomposition}
Sys
        Choice from
        Sys.T_f = 0  {System is free}
first
Convert_rule
        T_f     set     3       {Set of the counter
flag}
        Pz_c    set     1
$End
```



Step : 100

Resource : 612

Number : 100

Fig.6. Example of an animation frame

As performance measures, received at imitation are used such as: average time of processing of an request of IA on reception of a demanded resource, average load Channel IA, average load of the Serv-

ice agent and other.
The cell of sawmill shop was simulated. Basic animation frame is shown in Fig.6.

Conclusion

The considered approach to multi-agent model of resources exchanging ensures the decentralized production management in CIM. It is applicable to management systems incorporating specially developed LMS, which are capable to realize the inference mechanism, to receive, to accumulate and to treat knowledge.

Bibliography

[Artiba et al.1998] Artiba A., Emelyanov V.V., Iassinovski S.I. Introduction to Intelligent Simulation: The RAO Language. Kluwer Academic Publishers. 1998.
[Artificial 1996] Artificial life/Ed. by C.G.Langton and T.Shimohara.- Cambridge MA: MIT Press, 1996.
[Attoui et al., 1995] Attoui A., Hasbani A., Maouche A. Specification environment for multi-agent systems based on anonymous communications in the CIM context., Proc. ofIMSE, European Workshop on Integrated Manufacturing Systems Engineering, 12-14 December, Grenoble, (1995) pp. 243-252.
[Baptiste et al., 1993] Baptiste, P., Manier, H. Pilotage d'un anneau flexible de production: une approche definissant un comportement autonome par station. Proc. of GSI4: Quatrieme Congres International de Genie Industriel, Marseilles, 15-17 December, 1993. P. 117-126.
[Barbuceanu et al., 1994;] Barbuceanu, M., Fox, M.S. The information agent: an infrastructure agent supporting collaborative enterprise architectures. Proc. of 3rd Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Morgantown, WV, USA, 1994. P.112-116.
[Braspenning 1997] Braspenning P.J. Plant-like, Animal-like and Humanoid Agents and Corresponding Multi-Agent Systems. Proc. of the Int. Workshop "Distributed Artificial Intelligence and Multi-Agent Systems" (DAIMAS'97). St. Petersburg, Russia. 1997. P.64-77.
[Chandra et al., 1991] Chandra, J., Talavage, J. Intelligent dispatching for flexible manufacturing. International Journal of Production Research, 29, 1991. P.2259-2278.
[Emelyanov 1990] Emelyanov V.V. Algorithms of centralized and decenralized management systems of manufacturing cell. Moscow. The BMSTU Press, 1990 (in Russian).
[Emelyanov 1997] Emelyanov V.V. Multi-Agent Model for Manufacturing Systems Decentralized Control. Proc. of the Int. Workshop "Distributed Artificial Intelligence and Multi-Agent Systems" (DAIMAS'97). St. Petersburg, Russia. 1997, P.294-303.
[Emelyanov et al.,1997] Emelyanov V.V., Iassinovski S.I. An AI-based object-oriented tool for discrete manufacturing systems simulation. Journal of Intelligent Manufacturing. Vol.8, №1, February 1997. P.49-59.
[Fleury et al. 1984] Fleury G., Goujon J.-Y., Gourgand M., Lacomme P. Multi-agent simulation for planification of manufacturing systems. Computer Integrated Manufacturing and Automation Technology. CIMAT'96. Grenoble, France, May 29-31, 1996. P.148-153.
[Gornev et al. 1990] Gornev V.F., Emelyanov V.V., Ovsiannikov M.V. Management of Flexible Manufacturing Systems. Moscow, Mashinostroenie, 1990 (in Russian).
[Iffnecker et al., 1991] Iffhecker, C., Ferber, J., Zawadzki, D. Un modele multi-agents pour l'aide a la cooperation de produits elect-romecaniques, in Proceedings of Onziemes Jownees Internationales les Systemes Experts et Lews Applications, Conference Generate Systemes Experts de Seconde Generation, Ed. EC2, Vol 2, Avignon, May, 1991. P. 137-151.
[Jennings 1995] Jennings N.R. Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions. Artificial Intelligence, 75 (2), 1995. P. 195-240.
[Kouiss 1997]. Kouiss K., Pierreval H.,Mebarki N. Using multi-agent architecture in FMS for dynamic scheduling Journal of Intelligent Manufacturing. Vol.8, №1, February 1997. P.41-47.
[Kwok et al., 1993] Kwok A., Norrie D. Intelligent agent systems for manufacturing applications. Journal of Intelligent Manufacturing, 4, 1993. P.285-293.
[Larry 1995] Larry R. Medsker. Hybrid Intelligent Systems. Kluwer Academic Publishers, 1995.
[Lefrancois et al., 1994] Lefrancois P., Montreuil B. An object-oriented knowledge representation for intelligent control of manufacturing workstation. IIE Transactions, Industrial Engineering Research and Development, 26, (1), 1994. P.11-26.
[Parunak 1993] Parunak H.D. Industrial applications of multi-agent systems. Proc. Of INFAUTOM'93: Du Traitement Reparti aux Systemes Multi-Agents ei a l'Autonomie des Systemes', 18-19 February, Toulouse, Session B3 P.16.
[Tacquard et al., 1994] Tacquard C., Baptiste P., Manier H. Model of a behavior approach to schedule and control a flexible manufacturing ring: an extension to the pull flow. Proc. of IMSE, European Workshop on Integrated Manufacturing Systems Engineering, 12-14 December, Grenoble, 1994. P.333-339.
[Tarasov 1996] Tarasov V.B. Artificial life as a framework for enterprise modeling and reengineering. Proc. of IMACS Multi-conference "Computational Engineering in Systems Applications (CESA'96)", vol.4 "Symposium on Robotics and Cybernetics" (Lille, France, July 9-12, 1996). 1996, P.903-908.
[Trentesaux et al., 1995] Trentesaux D., Tahon C. Dynamic and distributed production activity control: a multicriteria approach for task allocation problematic. Proceedings of IEPM'95 International Conference on Industrial Engineering and Production Management, Marrakech, April, 1995. P.137-155.
[Varshavskiy et al. 1984] Varshavskiy V.I., Pospelov D.A. Orkestr igraet bes dirigera. Moscow, Nauka.1984 (in Russia).
[Velasco et al. 1995] Velasco J. R., Gonzalez J. C., Iglesias., C. A., Magdalena L. Multiagent based control systems: a hybrid approach to distributed process control. In A.E.K. Sahraoui and J.A. de la Puente, editors, Preprints of the 13th IFAC Workshop on Distributed Computer Control Systems, DCCS-95, pages 7-12, Toulouse, France, September 1995.
[Virdhagriswaran 1995] Virdhagriswaran S. Heterogeneous information systems integration - an agent messaging based approach. In Proceedings of the Third international Conference on Information and Knowledge Management (CIKM'94), November 1994.
[Zweben 1997] Zweben M. Intelligent Agents, Computer integrated manufacture and engineering (CiME), Vol.1, № 1, 1997. P.14-15.

REASONING WITH COMMON KNOWLEDGE AND SELF-REFERENCE IN MULTI-AGENT SYSTEMS

Maria Fasli[1]

[1]*University of Essex, Department of Computer Science, Wivenhoe Park, Colchester CO4 3SQ, UK*
email:mfasli@essex.ac.uk

Abstract

The study of intelligent agents has received an increasing attention within many disciplines. In this paper we argue about the importance of common knowledge and the need to be able to express self-referential statements in multi-agent systems. Based on these observations we present a first order self-referential framework for reasoning about truth, knowledge, and common knowledge. We continue by discussing a special case, a well-known logical paradox the surprise examination, which involves self-reference in a multi-agent domain, and we employ common knowledge to investigate it under an entirely new perspective.

Keywords: *Agent Theories, Common Knowledge, Logics in AI.*

1. Introduction

The term autonomous agent is usually employed to describe systems that are capable of independent action and rational behaviour in an open, and often unpredictable environment. Therefore they often need to communicate, co-ordinate and collaborate with one another in order to achieve a mutual goal, perform a complex task or share resources. Here we are viewing agents from the standpoint of Computer Science and we adopt a mentalistic view for them, we define agents as having certain mental qualities such as knowledge and beliefs and thus we characterize their behaviour in terms of these mental attitudes. This approach of ascribing human properties and attitudes to artificial agents, known as the intentional stance [2], is a useful and convenient means of describing complex systems as well as, explaining and predicting their behaviour. Using the intentional notions we can formulate agent theories in order to describe an agent, its properties and its reasoning. Hence a theory of agents can be viewed as a specification language which can be used to design, build, study and verify multi-agent systems.

A number of formalisms have been proposed in the literature for describing agents [3,9]. However most of them are based upon classical propositional or first-order modal logic. Using modal logics and possible worlds semantics is an attractive way for formalizing notions such as knowledge or belief but this approach lacks the characteristic of self-reference. Higher order logics and syntactic theories are possible avenues for incorporating self-reference but they have their own disadvantages like undecidability in the former case and inconsistency in the latter. If we want to obtain consistent syntactic theories this means weakening the logics but we end up with systems that are too weak to work with. On the other hand if we try to implement self-reference in modal logics the resulting systems suffer from the same drawbacks as syntactic theories [9].

A basic notion that should be built-in into an agent's cognitive system is of course that of truth. Therefore, the first requirement that emerges for a theory of agents is that it should be able to represent facts about the agent's world and obviously their relation to truth. An agent that can represent certain facts about its environment should also be aware of them, know or believe them.

For agents to interact not only with their environment but with the other agents as well, their theory should involve reasoning about the other agents and their knowledge, beliefs, preferences, goals, etc. Thus the second requirement that emerges for such a theory of agents is that it should be able to express self-referential statements. This stems from the fact that an agent's assertions about the other agents' knowledge may be self-referential. Self-reference occurs when an agent knows or believes a proposition about the knowledge or beliefs of other agents, which in turn make reference to this very proposition.

The third issue that concerns us, since we are interested in agents that interact with other agents in a number of ways, is the representation of the common knowledge that arises among a group of agents. The concept of common knowledge is based on what everyone in a group of agents knows and was first studied in the context of conventions [7] and has also attracted considerable attention in Economics [1] and Game Theory. However, it seems essential in co-ordination, collaboration, reaching agreements and in discourse understanding. Some of these issues will be discussed in the sequel.

This paper contributes towards the direction of reasoning agents in a number of ways. Firstly an alternative logical framework with possible worlds for describing agents is presented. We provide a first-order self-referential approach for reasoning about truth and knowledge, in a multi-agent domain working and extending the framework for syntactic modalities presented in [12]. In our approach we allow multiple syntactic modalities for knowledge since we want the theory to be applied in a multi-agent domain. Secondly we formalize the notion of common knowledge and its properties, which we argue is crucial for an agent's social behaviour and interaction. We discuss a special case, a well-known logical paradox, the surprise examination which involves a multi-agent domain and self-reference. After presenting the traditional analysis of the surprise examination we investigate it under an entirely new perspective using common knowledge.

The structure of the paper, which consists of five subsequent sections, is as follows. In the next section we briefly discuss the basic ingredients of an agent's cognitive system. We then informally establish the necessity for common knowledge in a multi-agent system and its effects on an agent's social behaviour and interaction. The following section describes the basic logical machinery that supports the notions of truth, modal knowledge and common knowledge and then we look at an extension of the framework in which knowledge and common knowledge are treated as syntactic modalities. We continue by discussing a special case, the surprise examination, using common knowledge as the new means of investigating it. The last section is a summary and a brief description of our findings, conclusions and a pointer to future work.

2. Truth and Self-Reference

The agents considered in this paper are viewed as systems that are attributed mental states such as knowledge and beliefs. These mental states are propositional attitudes, relations between an agent and various propositions about the world. A general theory for the representation of such attitudes should obviously represent truth and falsity. For example we would like to say that the predicate "true" applies to the quoted form of a sentence just if the sentence is true. That is, true("couple(zoe,john)") should be true if couple(zoe,john) is true. The Tarskian Biconditional captures this intuition:

$T(\phi) \Leftrightarrow \phi$

But this definition is self-referential, it includes reference to itself and unfortunately self-reference whether it is used directly or indirectly can lead to inconsistencies. We can construct a sentence, which asserts its own falsehood like:

ϕ. Sentence ϕ is false (The "Liar" sentence)

which leads to a vicious circle. To see why, we should be able to say either that it is false that sentence ϕ is false or that it is true that sentence ϕ is false. But if it is true that sentence ϕ is false then sentence ϕ is false. And if it is false that sentence ϕ is false then sentence ϕ is true. Reasoning agents should be able to make assertions about their knowledge and beliefs as well as about other agents' knowledge and beliefs. These beliefs or knowledge may involve quantification over sentences such as in "John believes something false". In other cases an agent's A assertion may involves a proposition about the beliefs of another agent B whose beliefs make reference to the very first proposition asserted by the first agent A, as for instance in the following case (indirect self-reference):

A : "I believe whatever B believes"

B : "I have the same beliefs as A"

Therefore, if statements like the above are to be represented, an adequate formal treatment of belief or knowledge must be able to refer to statements about the world as objects in the world. But quantification over propositional attitudes and self-referential sentences are beyond the expressive power of first order logic and can be expressed either in higher order logics or by using syntactic theories. But as was mentioned earlier both higher order logics and syntactic theories have disadvantages. Therefore we need a self-referential approach that will not run into problems and will be expressive enough for describing our intuitions about the intentional notions.

3. Common Knowledge in MAS

In this section we will discuss in a few more details the concept of common knowledge and we will argue about its need in a multi-agent environment. Recall from the discussion above that common knowledge is based on what everyone in a group of agents knows. For instance everyone in our society knows (ideally) that a green light means go and a red light means stop [3]. Moreover, this fact is also common knowledge, because not only everyone knows this fact, but everyone knows that everyone else knows this fact, and everyone knows that everyone knows that every individual knows this fact, and so on. Common Knowledge was first discussed by [7] in his philosophical study of conventions. Although we are not going to engage in a sociological analysis of conventions and how they are formed, nevertheless we have to mention here that conventions and social norms are regularities in behaviour. A convention is nothing more than a group prescription to do or not to do given actions under certain circumstances, a co-operative social behaviour and everyone that belongs to a certain group or community has to conform to it. What is important in the formation of conventions and social norms however, is the concept of common knowledge. In order for something to be a convention among the members of a certain group, everyone must be aware of the conditions of the convention, and everyone must know that every other member of the group knows that and that everyone is willing to conform to this convention. In other words everyone in the group must have common knowledge of the convention. Structured groups of human agents like societies, communities and organisations, operate according to certain predetermined authority relationships and social norms. Once an agent enters into a community he is given a specific role which entails a set of rights and obligations. Each member of the group knows its place and acts accordingly and furthermore each knows the implications of exercising rights and braking commitments. These authority mechanisms rely on the obedience of the addressees and their behaviour, and enable the whole group to act effectively and efficiently. In such a structured community the notion of common knowledge naturally arises. For instance, the leader of such a group has a certain predetermined authority and this in fact is common knowledge among all the members of the group. The members of the group also know that they are obliged to carry out certain tasks and obey the leader and this is profoundly common knowledge. However it seems reasonable to suggest that agents in artificial societies or in multi-agent systems with build-in or with the ability to form authority relationships and communities or groups, should have common knowledge of the whole structure as well of their role in it and the rights and obligations regarding themselves and the other members of the group.

Other forms of social commitment as promises for example include the element of common knowledge as well. We usually prefer to keep our promises and this is a moral obligation that we undertake when we make one. We keep our promises because we do not want to destroy our reputation for not keeping promises, lose other people's trust or undermine trust and confidence in promises in general. It is common knowledge that a promise indicates a person's true intentions and his/her commitment in carrying it out. From now on and throughout this paper we will assume that we are dealing with rational, truthful and sincere agents, who have no intention of deceiving one another and do not, on purpose, supply false information.

An ordinary way of acquiring common knowledge is through communication. Announcements made in public are sources of common knowledge. The purpose of the speaker is to make her intentions, goals, preferences etc. known to a group of other agents. In other words to provide knowledge not only to each member of the group separately but to the group as a whole, that is she provides common knowledge. The content of the announcement can then be used by each of the members individually and as a group as well, in order to make decisions, plans or take action. Simple speech acts between two or more agents; exchanging information can be the source of common knowledge as well, on the grounds that the agent supplying common knowledge is trustworthy and reliable.

Common knowledge is also relevant in discourse understanding and in the use of language among a population. Grammar rules and word meaning are considered common knowledge among a population using a specific language. Consider the case of a new expression added to the language. In order to use this expression a group must have common knowledge of it, how it is used, what it means and that everyone knows how to use it as well. In a number of applications involving agents such as negotiation and agreements, co-ordination and collaboration, common knowledge seems to be relevant as well. For instance suppose that two agents A and B need to agree on some statement x. It is reasonable to suggest that if the two agents agree on x then each of them knows that they have agreed on x. As was pointed out in [3] this is a key property of agreement: each of the agents must know about the agreement and he must also know that every other participant in the agreement knows about the agreement. Thus an agreement always presupposes common knowledge among all the agents involved. In co-ordination situations the agents need common knowledge for example of the schedule of actions to be performed by each agent. In cases such as the coordinated attack where common knowledge cannot be attained, co-ordination becomes difficult if not at all impossible [3].

But how does common knowledge affect an agent's social behaviour and influences an agent's decisions, choices, future actions and goals? For instance, in the case when an agent belongs to a certain structured group, community or organization, the agent has to adopt social commitments and must undertake obligations and rights. But a socially committed agent to a group or another agent, loses some of its autonomy in order to adapt to the organizational constraints. Therefore an agent having now common knowledge of certain restrictions may need to modify his behaviour to conform to these restrictions. His goals and future plans and actions may be in conflict with the constraints and requirements of the group and thus may need modifications and alterations. Maybe some of them need to be abandoned completely. Common knowledge is not an abstract definition but a key concept, a basic and active ingredient of an agent's reasoning process and affects and guides the agent's social behaviour.

4. Logical Framework

In this section we present an alternative possible worlds formalism for agents. Our logical machinery is based on the framework proposed in [12]. We build on this work and we extend it further by adopting the approach of [3] towards common knowledge.

4.1. The Logical Language

We begin by presenting the logical language we will be using to write down our logics. Our language L is based on First Order Language and apart from the standard connectives and quantifiers it also includes:

i. Three distinctive predicate symbols T, F and = for expressing Truth, Falsity and Equality respectively.
ii. Three modal operators K_i, E_G and C_G for expressing "Agent i knows", "Everyone in a group G knows" and "It is common knowledge among the members of G" respectively.
iii. A set of variable symbols V and a set of predicate symbols P.

Terms in this language are:

i. Constants .
ii. Variables
iii. wffs can be treated as terms in the language in order to allow circular reference. Thus if $A(y_1, y_2,..,y_n)$ is a wff and $y_1, y_2,..,y_n$ are free variables then $(t_A, y_1, y_2,..,y_n)$ is a term. (In what follows we write A where no confusion can arise.)

An atomic formula is a predicate letter applied to terms; if P is a predicate letter and $t_1, t_2,..,t_n$ are terms the $P(t_1, t_2,..,t_n)$ is an atomic formula. The formulas of the language (wffs) are then defined inductively as follows:

i. An atomic formula is a wff
ii. If t_1 and t_2 are terms then $t_1=t_2$ is a wff
iii. If t is a term then F(t) and T(t) are wffs
iv. If A and B are wffs so are $\neg A$, $A \wedge B$, $A \vee B$, $A \Rightarrow B$ and $A \Leftrightarrow B$.
v. If x is a variable and A is a wff then $\forall x A$ and $\exists x A$ are wffs.
vi. if A is a wff then $K_i(A)$, $E_G(A)$ and $C_G(A)$ are wffs

A model for the logical language L is a tuple $M=<W, K_i, D, \pi, T, F>$ where W is a set of possible worlds, K_i is the accessibility relation for each agent i of our multi-agent domain; D is the universe of discourse; π is used to determine the truth values of the atomic formulas of the language apart from the truth and falsity predicates; T is the extension of the truth predicate (that is T:D×W$\Rightarrow$ 0,1) and F is similarly the extension of the false predicate. We require the domain of individuals to be constant and Cartesian closed and each individual constant to be a rigid designator.

4.2. Stable Truth

Our intuitions for the truth predicate tell us that whatever is asserted to be true, must be so and this is elegantly captured by the Tarskian Biconditional:

Tb. $T(A) \Leftrightarrow A$

However, as we mentioned earlier the inclusion of such an axiom schema in our logic leads to inconsistency. In order to maintain consistency the logic must be weakened. In the Gupta-Herzberger semantic theory this is achieved by discarding the principle of bivalence:

Biv. $T(A) \vee T(\neg A)$

and this in other words means that not all sentences in the language denote propositions, some sentences are paradoxical.

The intuitions underlying the Gupta-Herzberger [4,5] semantic theory are based on the idea of an iterative revision process. This revision process starts with simple statements that do not contain the word true and they are assigned a truth value according to the empirical facts. As the process continues, more and more statements involving complex assertions about truth and falsity are assigned a truth value. Given the model M for L, as defined above, we define $M'=<W, K_i, D, \pi, T', F'>$ the Tarskian revision of M such that:

$T'=(||t||)_{v,w}=$ (a) 1 iff $t=(t_A, y_1,...y_n)$ and $M \models_{v,w} t$
(b) $T'=(||t||)_{v,w}$ otherwise

$F'=(||t||)_{v,w}=$ (a) 1 iff $t=(t_A, y_1,...y_n)$ and $M \models_{v,w} \neg t$
(b) $F'=(||t||)_{v,w}$ otherwise

where v is an assignment of elements of D to variables. Only the wff change since for the other elements such as constants and variables the T and F do not change. Starting from a model M with arbitrary extensions of T and F and using the above revision step, we can define a sequence of Truth and Falsity predicates T(n) and F(n) for n>=0 :

i. $T(0) = T$
$F(0) = F$
ii. $T(n+1) = T(n')$

$F(n+1) = F(n')$

iii. for a limit ordinal k define:

$T(k)(d)=1$ iff $\exists\ j(j<k)\ \forall\ h(j<=h<k)\ (T(h)(d)=1)$

$F(k)(d)=1$ iff $\exists\ j(j<k)\ \forall\ h(j<=h<k)\ (F(h)(d)=1)$

Under this revision process the notion of stability can be defined as follows:

- an element d of D is positively stable iff $\exists$ j $\forall$ k>=j $T(k)(d) = 1$
- an element d of D is negatively stable iff $\exists$ j $\forall$ k>=j $F(k)(d) = 1$
- d is stable iff it is positively or negatively stable
- d is positively stable from k iff $\forall$ j>=k $T(j)(d)=1$
- d is negatively stable from k iff $\forall$ j>=k $F(j)(d)=1$

The notion of a stabilization ordinal is central to the Gupta-Herzberger approach. A stabilization ordinal is that point in the revision process at which no more objects will become (stably) true or false, that is we have reached a saturation point where we can say no more about truth and falsity. An ordinal σ is a stabilization ordinal iff:

i. $\forall d \in D$, d is positively stable iff $T(\sigma)(d) = 1$
ii. $\forall d \in D$, d is negatively stable iff $F(\sigma)(d) = 1$
iii. $\forall d \in D$, d is positively (negatively) stable implies that d is positively (negatively) stable from σ.

THEOREM (Herzberger) There exists a stabilization ordinal.

We are interested in wffs that are valid in such stabilized models as described above. Thus we have the following three additional definitions:

i. a wff A is *safe* iff A is valid at every stabilization ordinal
ii. A is *stably true* iff T(A) is safe and
iii. A is *stably false* iff F(A) is safe

The models with which we are going to be concerned here are the D, T, S4 and S5 models. In these models the accessibility relation among possible worlds is serial for D models, reflexive for T models, transitive and reflexive for S4 models, and finally transitive, symmetric and reflexive for S5 models. We are going to refer to these models from now on as Γ-models. The notion of safeness and stability can be relativized to the class of Γ-models under consideration. A wff is Γ-safe iff A is valid at all stabilization models for all initial Γ-models M. It is Γ-positively stable if T(A) is Γ-safe and it is Γ-negatively stable if F(A) is Γ -safe.

4.3. Logics of Truth, and Modality

The weakest logic that we can have for truth, modal knowledge and common knowledge is D_{Tm}:

k. $T(A \Rightarrow B) \Rightarrow (T(A) \Rightarrow T(B))$

d. $T(A) \Rightarrow \neg T(\neg A)$

bar. $\forall x T(A) \Rightarrow T(\forall x A)$

Nec If $D_{Tm} \vdash$ then $D_{Tm} \vdash T(A)$

K. $K(A \Rightarrow B) \Rightarrow (K(A) \Rightarrow K(B))$

D. $K(A) \Rightarrow \neg K(\neg A)$

BAR. $\forall x\ K(A) \Rightarrow K(\forall x A)$

NEC If $D_{Tm} \vdash$ A then $D_{Tm} \vdash K(A)$

E. $E_G(A) \Leftrightarrow \wedge_{i \in G} K_i(A)$

C. $C_G(A) \Leftrightarrow E_G^k(A)$ for k=1,2,...

IR. if $A \Rightarrow E_G(A \wedge B)$ then $A \Rightarrow C_G(B)$

We have chosen the standard K- and D-axioms for both truth and knowledge. K is the minimal system for normal modal logics [6] and it states that if an agent knows A and knows that A⇒B then she also knows B. The D-axiom expresses the consistency of an agent's knowledge and the necessitation rule states that any valid formula is known. The necessitation rule and the K-axiom are forced on us by the possible worlds framework itself and give rise to the logical omniscience problem.

We follow the approach of [3] on Common Knowledge. Intuitively everybody in a group knows a formula A if and only if every agent *i* in that group knows A. As we mentioned earlier if A is common knowledge among a group then everyone knows it and everyone knows that everyone else knows it, and everyone knows that everyone knows that everyone else knows it, and so on. This notion of common knowledge among a group of agents has a graph interpretation and requires the notion of reachability, which involves paths of arbitrary finite length. A group of agents G has common knowledge of A in a world w if there is a world w' such that there is a path in the graph from w to w' whose edges are labeled by members of G (w' is G-reachable from w) [3].

THEOREM 1.
D_{Tm} is a consistent logic of truth and modality and all the theorems of D_{Tm} are stably true.

Proof.
In order to establish that each of the theorems of D_{Tm} is stably true, we first establish that each of the axioms are not only safe (axiom k for instance is true at stabilization ordinals) but stably true (that is, T(k-

axiom) is true at stabilization ordinals). We will illustrate with the axioms of truth.

(1) $T(T(A \Rightarrow \neg T(\neg A))$

(2) $T(T(A \Rightarrow B) \Rightarrow (T(A) \Rightarrow T(B)))$

(3) $T(\forall x T(A) \Rightarrow T(\forall x A))$

For (1) we have only to observe that T(A) always excludes the possibility of $T(\neg A)$ at both successor ordinals and limits. For (2) we must show that $(T(T(A \Rightarrow B) \Rightarrow (T(A) \Rightarrow T(B))$ is true at any stabilization ordinal. First note that $T(A \Rightarrow B) \Rightarrow (T(A) \Rightarrow T(B))$ is true at any successor ordinal by the definition of revision. Moreover at limit ordinals if $A \Rightarrow B$ has been true from some ordinal less than the limit ordinal as well as A, then B must have been true from the greater of the two ordinals and thus is true at the limit. The argument for the safeness of the Barcan formula (3) relies on the constancy of the domain of individuals. For the Nec rule for truth we have only to observe that if A is true at every world in every model from some ordinal onwards then T(A) will be.

For the modal case we will only illustrate with the K-axiom. Suppose that $M_{v,w} \models K_i(A \Rightarrow B)$ and $M_{v,w} \models K_i(A)$, then there is a world w' such that $K_i(w,w')$ in which $M_{v,w'} \models (A \Rightarrow B)$ and $M_{v,w'} \models (A)$. Then using modus ponens it must be the case that $M_{v,w'} \models (B)$. Since $K_i(w,w')$, it follows that $M_{v,w} \models K_i(B)$.

For common knowledge it is quite easy to establish that the axioms are safe since they are true at all worlds in the D-models.

•

We would like to strengthen the axiomatization for truth by considering stronger axioms like t, s4, and s5:

t. $T(A) \Rightarrow A$

s4. $T(A) \Rightarrow T(T(A))$

s5. $\neg T(A) \Rightarrow T(\neg T(A))$

The t-axiom is the axiom of truth, saying that whatever is asserted to be true must be so. The s4- and s5-axioms state that if a fact is true then it is true that it is true, and if a fact is false then it is true that it is false. However, as Turner [12] shows the incorporation of these axioms to the logic of truth results in inconsistency. The t and s4 schemata are safe but not stable, whereas the s5 schema is not even safe. The only way to incorporate the axioms t and s4 to the logic is by weakening the necessitation rule in order to obtain consistency.

In order to strengthen the modal part of the logic we can add the standard modal axioms for knowledge to which we are going to refer as the set Y:

T. $K(A) \Rightarrow A$

S4. $K(A) \Rightarrow K(K(A))$

S5. $\neg K(A) \Rightarrow K(\neg K(A))$

The T-axiom is the axiom that separates philosophically the notion of knowledge from that of belief. Knowledge is always true whereas an agent can have false beliefs (without of course being aware of it). The axioms S4 and S5 are the positive and negative introspection axioms respectively and give the agent introspective capabilities with regards to her knowledge.

Let us call D[X,Y] the logic which comprises of the aforementioned D_{Tm} logic and the sets of axioms Y and X together with the rules:

Neca If $D[Y] \vdash A$ then $D[X,Y] \vdash T(A)$

Necb If $D[X,Y] \vdash A$ then $D[X,Y] \vdash K_i(A)$

THEOREM 2.

D[X,Y] logics are consistent systems of truth and modal knowledge and common knowledge.

Proof.

The proof of this theorem relies on the proof of THEOREM 1. For the rest of the axioms we follow a similar line. Consider the Necb rule. If A is true at every world for every stabilization ordinal then $K_i(A)$ will be. The rule Neca for truth follows from the fact that all the theorems of D[Y] are Γ-stable where Y is any combination of modal axioms for knowledge.

•

4.4. Extending the Framework

Let us extend the language L to L1 by adding three new predicates $KNOW_i$, EK_G and CK_G for expressing knowledge, what everyone in a group of agents knows and what is common knowledge in a group of agents respectively. Since it is well known that a direct treatment of knowledge as a predicate can result in inconsistency, we are going to use an alternative approach here in order to obtain consistency on the one hand, and strong enough logics on the other. In this approach, following [12] the knowledge predicate is defined as follows:

$KNOW_i(A) =_{def} K_i(T(A))$

Now the central question is what are the available logics to us for Truth and syntactic knowledge given the above definition. The weakest logic is D_{TK}:

k. $T(A \Rightarrow B) \Rightarrow (T(A) \Rightarrow T(B))$

d. $T(A) \Rightarrow \neg T(\neg A)$

bar. $\forall x T(A) \Rightarrow T(\forall x A)$

Nec if $D_{TK} \vdash A$ then $D_{TK} \vdash T(A)$

K. $KNOW_i(A \Rightarrow B) \Rightarrow (KNOW_i(A) \Rightarrow KNOW_i(B))$

D. $KNOW_i(A) \Rightarrow \neg KNOW_i(\neg A)$

BAR $\forall x$ $KNOW_i(A) \Rightarrow KNOW_i(\forall xA)$

NEC If $D_{TK} \vdash A$ then $D_{TK} \vdash KNOW_i(A)$

THEOREM 3
D_{TK} is a consistent logic of truth and syntactic modality. Now again if we try to strengthen the logic we can get the following family of consistent logics.
Proof.
In order to show consistency we employ the definition of the modal predicate. We will illustrate with the D-axiom for knowledge. Hence we get $\neg(KNOW_i(A) \wedge KNOW_i(\neg A))$ and by the translation $\neg(K_i(T(A)) \wedge K_i(T(\neg A)))$. Using the d-axiom for truth we get $T(\neg A) \Rightarrow \neg T(A)$ and from the necessitation rule for K_i we get $K_i(T(\neg A) \Rightarrow \neg T(A))$. From the assumption and the K-axiom for K_i we can deduce $K_i(\neg T(A))$. But now we have $K_i(T(A)) \wedge K_i(\neg T(A))$ which contradicts the K-axiom for K_i. The other two axioms and the rule follow in a similar way.

•

Let X be any subset of the standard axioms for truth t,s4:

t. $T(A) \Rightarrow A$

s4. $T(A) \Rightarrow T(T(A))$

Let Y any subset of the standard modal axioms for knowledge T,S4 given below:

T. $KNOW_i(A) \Rightarrow A$

S4. $KNOW_i(A) \Rightarrow KNOW_i(KNOW_i(A))$

Let us call D[X,Y] the logic which comprises of the aforementioned D_{Tm} logic and the sets of axioms Y and X together with the rules:

NecA If $D_{TK} \vdash A$ then $D_{TK}[X,Y] \vdash T(A)$

NecB If $D_{TK} \vdash A$ then $D_{TK}[X,Y] \vdash KNOW_i(A)$

THEOREM 4
$D_{TK}[X,Y]$ logics are consistent systems of truth and modal knowledge and common knowledge.
Proof.
The proof follows in a similar way like the proof of THEOREM 3, by employing the translation of the syntactic modalities.

•

The following axiom connecting truth and knowledge can be consistently added to the aforementioned logics:

$T(KNOW(_i(A)) \Leftrightarrow KNOW_i(T(A))$

The predicate EK_G can be defined as a syntactic modality as follows:

$EK_G(A) =_{def} E_G(T(A))$

The predicate for common knowledge now is similarly defined as follows:

$CK_G(A) =_{def} C_G(T(A))$

The properties of these new predicates reflect the properties of the respective modal operators:

E. $EK_G(A) \Rightarrow \bigwedge_{i \in G} KNOW_i(A)$

C. $CK_G(A) \Rightarrow EK_G^k(A)$ for k=1,2,....

IR. if $A \Rightarrow EK_G(A \wedge B)$ then $A \Rightarrow CK_G(A)$

THEOREM 5
The $D_{TK}[X,Y]$ logics together with the above axioms for EK_G and CK_G predicates, are consistent logics of truth, syntactic knowledge and syntactic common knowledge.
Proof.
By employing the translation.

•

5. Common Knowledge and the Surprise Examination Paradox

In the preceding sections we mentioned the notion of self-reference and the fact that sometimes it can lead to inconsistencies. We will examine such a situation here, a logical paradox known as the surprise examination which involves self-reference and arises in a multi-agent environment. The situation can be told the following way:
The teacher tells the class that sometime during next week she will give an examination. She will not say on which day for, she says, it is to be a surprise. Is it possible for the teacher to give a surprise exam?
A student now can reason as follows. The exam cannot be delayed until Friday. Because if it hasn't taken place on any of the previous days then on Thursday the student will know that it will take place on Friday. If the student knows the day then it cannot be a surprise and that's why the exam day cannot be Friday. But the exam day cannot be Thursday either. Because if the exam hasn't taken place on any of the previous days and having already excluded Friday then the student thinks that the only possible day left is Thursday. But again if he knows the day then it cannot be a surprise and so the exam day cannot be Thursday. Using the same line of reasoning the student can eliminate the rest of the days as well. So finally he comes to the conclusion that there is not going to be an exam after all. But their teacher next week gives them the promised surprise examination. The paradox has been formalized in a number of ways in the literature and [11] contains a very thorough discussion. The teacher's announcement is indeed self-referential. She states on the one hand that there is going to be an exam some time next week and on the other that based on her very

announcement (this is where self-reference occurs, the teacher refers to her own announcement) the student will not know the day of the exam, it will come as a surprise. Although the surprise examination has been extensively studied from the point of view of self-reference, we believe that other important issues as well have not been sufficiently recognized and explored. As we will show in the sequel common knowledge seems to be the key element in this new investigation. In the surprise examination we have a group of agents which consists of a teacher and a class of students. Communication among any members of the group is achieved through announcements and speech acts, which is a way of making someone's plans and intentions known. Recall from the third section that the content of public announcements is considered to be common knowledge among the speaker and the audience after the announcement is made. Since the announcement is made in public this creates a moral obligation as well. The teacher's declaration of her intentions can be interpreted as a promise, a commitment to the students. If a promise is made in public (like the teacher's announcement) the others will know that they must go along with the one who has promised, for they know what she will do. If the teacher brakes her promise then she betrays the students' trust and faith in her, and it is difficult to think of an educational system in which the teachers cannot be trusted. In addition the surprise examination takes place among the members of a specific organized group, that of a class. In a class the students and the teacher have separate roles each with respective rights and obligations. Furthermore the teacher has a certain kind of authority which is well known and accepted among the other members of the group. Taking into account the whole setting of the surprise examination and from the discussion above we believe say that common knowledge arises in this case. A student belongs to a structured group, in which the teacher has an authoritative position which gives her the right to make decisions and the students are obliged to follow them. This is one reason for the student to take the teacher's announcement seriously. Moreover the teacher is considered to be truthful and sincere and this in fact is part of her obligations and commitments towards the class. Furthermore her intentions have taken the form of a promise, a strong social commitment and this is another reason for the student to believe the teacher's announcement. Going even further, if the student tries to model the teacher's reasoning he will be able to see that what the teacher promised to do is indeed achievable. Therefore it is quite reasonable for the student to conclude that having common knowledge of the teacher's announcement, he has common knowledge of the fact that he is not going to know the exact day of the exam, and this conclusion is supported by the above argumentation. A student's common knowledge and rationality provide him enough reasons so as to make the right choice. Obviously, in order to fully analyze the surprise examination we would need not only knowledge and common knowledge, but intentions, goals, commitments, obligations and actions. However here we are going to restrict our formal analysis to the use of common knowledge only, since our theory does not fully account for all these issues. The basic idea behind our formalization is that the self-referential announcement of the teacher does not include the kind of information necessary for the class to estimate the exact day of the exam. Therefore, knowing the announcement does not mean knowing the day of the exam, and this takes the following form:

$\phi{:}\phi_1\wedge(\mathrm{KNOW}(\phi)\Rightarrow\mathrm{KNOW}(\neg\psi))$

where

ϕ: "I will give you an exam next week and based on the knowledge of this announcement, you will not know the exact date of the exam",

ϕ_1 : "I will give you an exam next week",

ψ : "The exact date of the exam",

and ϕ_1 and ψ can be presented as first order formulas of our logical language. C is the class of students. Now we can show that with the introduction of common knowledge, the fact that the class will not know the exact day of the exam becomes common knowledge. This can be done with all the above mentioned logics of truth and syntactic modalities, but here we use D_{TK}. So formally we have:

1. ϕ assumption
2. $EK_C(\phi)$ everyone knows ϕ
3. $EK_C(\phi_1\wedge(\mathrm{KNOW}_C(\phi)\Rightarrow\neg\mathrm{KNOW}_C(\phi)))$ 2,definition of ϕ
4. $EK_C(\phi_1)\wedge EK_C(\mathrm{KNOW}_C(\phi)\Rightarrow\neg\mathrm{KNOW}_C(\psi))$, theorem of D_{TK}
5. $EK_C(\mathrm{KNOW}_C(\phi)\Rightarrow\neg\mathrm{KNOW}_C(\psi))$ 4
6. $EK_C(\mathrm{KNOW}_C(\phi))\Rightarrow EK_C(\neg\mathrm{KNOW}_C(\psi))$ 5, Theorem of D_{TK}
7. $CK_C(\mathrm{KNOW}_C(\phi))\Rightarrow CK_C(\neg\mathrm{KNOW}_C(\psi))$ The announcement is Common Knowledge

We conclude that common knowledge of the announcement implies that it is common knowledge that the class will not know the exact day of the exam. The class in fact is aware of their lack of knowledge of the examination day. This result is quite different from those found in the literature so far [11]. There is no good reason for rejecting the announcement as a false one since the teacher is considered to be truthful and sincere and has no intention of deceiving the class. It is not logical either to conclude that the class does not know the announcement. The fact that the announcement is made in public puts the students in a very special situation, in which they know that the announcement is true and that they are in this situation, the announcement has in fact become common knowledge.

6. Concluding Remarks

This paper has presented a method for the formal description of multi-agent systems, which combines the notions of truth, knowledge and common knowledge. Although, this work leaves many unanswered questions, we believe it is a first step towards the direction of a first-order theory of agents that ca reason about self-referential sentences. A first order language has been presented in which the concepts of truth, knowledge and common knowledge have been formalized based upon the framework for syntactic modalities set up in [12]. A number of properties have been introduced as axioms of a theory of agency concerning the aforementioned concepts. It is argued that the intuitions captured in this model provide a flexible way of describing agents and the kind of reasoning involved in a multi-agent environment. The framework is quite flexible and for instance, the weaker notion of belief can be formalized instead of knowledge. Our model presents the following advantages when compared with other approaches [3,10]. Firstly we use first order logic which offers more expressive power than classical modal logic. Furthermore, although the formal counterpart of first order theories, standard modal logics, offer another possible and attractive way for formalizing notions such as knowledge and belief, they lack the characteristic of self-reference. Once the mechanism for implementing circular reference is added to modal logic, the whole approach runs into problems and modal theories suffer from the same drawbacks as syntactic theories [9]. Our approach yields consistent self-referential logics for truth, knowledge and common knowledge, which are not too weak to work with. In addition in our framework statements like the "liar" and the "knower" [8] can be effectively blocked. In this paper we have also formalized the surprise examination based on our intuitions about the teacher's announcement as a self-referential statement, without however changing the meaning of the original statement, and finally we demonstrated how common knowledge can be employed in order to investigate it. Our example shows how an agent's choice and behaviour can be influenced and affected by considering social factors like conventions, authority relationships, group belonging and of course common knowledge. As it is the theory now, only accounts for the information component of an agent (knowledge) and says nothing about its motivational part such as intentions, goals or desires. In studying the behaviour of an agent this component is needed as well.

There are a number of possible avenues for future development of this model. Firstly the notion of time can be incorporated into the model and the relation between time, common knowledge and action can be studied. Secondly the model can be extended to a K(B)DI [10] model and enriched by incorporating intentions and desires as new syntactic modalities and such an approach is under current investigation. Desires and intentions could then be used in order to better analyze and comprehend the surprise examination. A third possible avenue for investigation is a higher order version of the approach presented here. However, while the ideas in this paper may be conceptually appealing, considerable work remains to be done to analyze the utility of the approach in more complex situations.

Acknowledgements

I would like to thank Ray Turner and the reviewers for their helpful comments on an earlier draft of this paper. The research reported here has been supported by GSSF grant 189/96.

Bibliography

1. Aumann R.J.(1976) Agreeing to Disagree. Annals of Statistics 4(6), pp.1236-1239
2. Dennet D.C. 1987. The Intentional Stance. Cambridge, Mass.: The MIT Press.
3. Fagin et al (1995) Reasoning about Knowledge. London, England : MIT Press
4. Gupta A. (1982) Truth and Paradox. J. of Philosophical logic, vol: 11, pp. 1-60
5. Herzberger H.(1982) Notes on Naive Semantics. J. of Philosophical logic, vol: 11, pp. 61-102
6. Hughes G. E. and Cresswell M.J. (1968). An Introduction to Modal Logic. London:Methuen.
7. Lewis D. (1969) Convention, A Philosophical Study. Harvard University Press, Cambridge, Massachusetts.
8. Montague R. and Kaplan D. A Paradox Regained. Notre Dame Journal of Symbolic Logic 1:79-90.
9. Perlis D. (1985) Languages with self-reference I: Foundations. Artificial Intelligence, Vol:25,pp. 301-322
10. Rao A. and Georgeff M. 1991. Modeling Rational Agents within a BDI-Architecture. In Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning. San Mateo, Calif.: Morgan Kaufmann Publishers.
11. Sainsbury R.M.(1995) Paradoxes. Cambridge University Press
12. Turner R. (1990) Truth and Modality for Knowledge Representation. The MIT Press, Cambridge Massachusetts

AGENT-BASED MODEL OF INFORMATION SECURITY SYSTEM: ARCHITECTURE AND FRAMEWORK FOR BEHAVIOR COORDINATION

Vladimir I. Gorodetski[1], Igor V. Kotenko[2], Leonard J. Popyack[3], Victor A. Skormin[4]

1 - St.-Petersburg Institute for Informatics and Automation. E-mail: gor@mail.iias.spb.su[1]
2 - St.-Petersburg Signal University. E-mail: ivkote@robotek.ru
3 - USAF Research Laboratory, Information Technology Division, E-mail:popyack@rl.af.mil
4 - Binghamton University, E-mail: vskormin@binghamton.edu

Abstract

The paper is focused on an agent-based information security system (ISS) application. Shown is an ISS which is network-based and distributed over a number of hosts of the computer network. Each host-based ISS consists of a number of interacting and cooperating autonomous agents managed by coordination agents. Agents are specialized for access control, detecting non-authorized intrusion, pursuing, identification and rendering harmless the attacker, accessing the damage of non-authorized access and information integrity recovery, authentication cryptographic defense, steganography and steganoanalysis. An agent-based model of a computer security system is proposed. It is based on ontology of the ISS domain that is the subject of development in the paper. Ontology is used as a means of structuring distributed ISS knowledge to help specify the common ground of interacting agents as well as for agents behavior coordination. General principles of agents' coordination within agent-based ISS are considered.

Keywords: *multi-agent system, information security, knowledge sharing, ontology, agent coordination, common ground.*

1 Introduction

The problem of information security is recognized now as one of the most complex and its importance is growing coherently with increasing network connectivity, size, and implementation of new information technologies. Today, information has become a highly valuable commodity and its vulnerability is of great concern within any large-scale organization utilizing computer networks. Networks and information are becoming increasingly vulnerable to intrusion due to new sophisticated threats and attacks, both direct and remote, aimed at overcoming or destroying existing information security means.

A widely accepted point of view intrusion is defined as "any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource" [4]. According to this definition, there exist three main types of threats for information security [3, 6, 7, 16]:

(1) threat of non-authorized access to information;

(2) threat of destroying information integrity, and

(3) threat of denial of service making crucial resource and/or information unavailable.

Currently used computer security systems consist of a number of independent components requiring an enormous amount of distributed and specialized knowledge to solve their own security sub-problems. As a rule, these systems represent a bottleneck with regard to process speed, reliability, flexibility and modularity [2]. A modern information security systems (ISS) must be considered as a number of independent, largely autonomous, network-based, specialized software agents operating in a coordinated and cooperative fashion designated to prevent particular kinds of threats and suppressing specific types of attacks. This can provide the required level of general security of information according to a global criterion.

A good and clear analogy has been drawn between agent-based ISS and the animal immune

[1] This author's work is supported partially by RFBR grant 99-01-00733

system [8]. The immune system consists of distributed white blood cells, which must attack anything which they suspect to be alien. By having "as many cells as necessary", the animal body is able to defend itself in a very efficient way. If the animal body is infected in one area, then cells move to that area and defend it.

Modern multi-agent system technology presents a valuable approach for the development of ISSs that is expected to have very promising advantages when implemented in a distributed large scale multi-purpose information system.

Here we consider an agent-based model of an ISS. The paper is organized as follows. In section 2, the conceptual level of such a model of an ISS is outlined. In sections 3 and 4, based on [3, 6, 7, 16, 17], we propose the ontology of an information security domain. It is considered as an important means to form the structure of a task - oriented distributed agents' knowledge and belief utilized for behavior coordination as well as a common ground for agent information exchange and their mutual understanding. In addition, ontology is considered as a basis for a general information security task decomposition and, hence, corresponding multi-agent-system architecture development. In section 5, we outline general principles of agents' negotiation and coordination within an agent-based ISS. Section 6 is devoted to analysis of related works associated with agent-based ISS and emphasizing distinctions of agent - based models of ISS proposed here. In conclusion we outline the main results and future work aimed at utilizing agent-based technology for ISS development.

2 Conceptual Agent-Based Model of ISS

Conceptually, a multi-agent ISS is viewed as a cooperative multitude of the following types of agents, distributed both across the network and on the host itself (see fig.1):

(1) *(discretionary and mandatory) access control agents* which constrain access to the information according to the legal rights of particular users;

(2) *audit and detecting non-authorized access agents (intrusion detection agents)* and to alert a responsible system (agent) about potential occurrence of a security violation;

(3) *anti-intrusion agents* responsible for pursuing, identifying and rendering harmless the attacker;

(4) *agents of accessing the damage of non-authorized access and information integrity recovery*;

(5) *cryptographic defense agents*;

(6) *steganography and steganoanalysis agents*;

(7) *authentication agents* responsible for identification of the source of information, whether its security was provided during the transmission which provide verification of identity.

Coordinated and cooperated behavior of the above agents provides the required level of general security of information according to a global criteria which are managed by agents of a special type which, according to the accepted terminology, are called *meta-agents*, or, simply, *managers*.

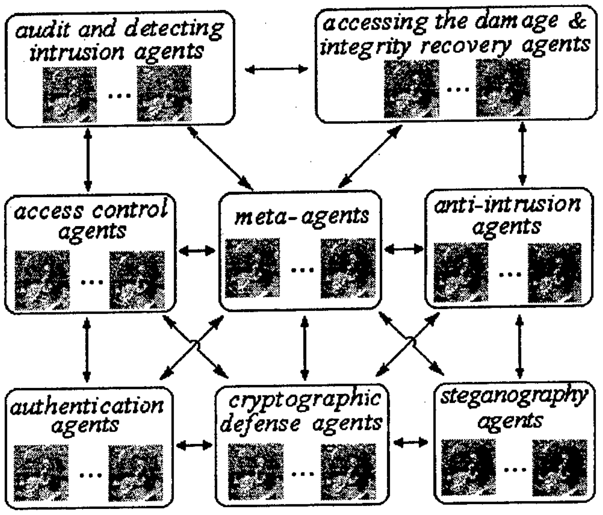

audit and detecting intrusion agents
accessing the damage & integrity recovery agents
access control agents
meta-agents
anti-intrusion agents
authentication agents
cryptographic defense agents
steganography agents

Fig.1. Structure of security agents' community

According to the proposed conceptual multi-agent model of information security system, let us specify processes of performance for protected information systems as association of three functions: $Func = Func^1 \cup Func^2 \cup Func^3$. $Func^1 = N^1 \cup M^1$, $Func^2 = N^2 \cup M^2$, $Func^3 = N^3 \cup M^3$, where $N^1: T \rightarrow C$ and $M^1: T \rightarrow C$ - are the processes of messaging on the authorized access channels and the non-authorized access channels respectively, $N^2: T \rightarrow C$ and $M^2: T \rightarrow C$ - are the processes of messaging causing and not causing violation respectively, $N^3: T \rightarrow C$ and $M^3: T \rightarrow C$ - are the processes of messaging not resulting and resulting to denial of service respectively, T - is the set of discrete count of time, C - is the set of the messages generated during the normal work of the protected information systems.

Let us determine a general task of information security as a task of finding such security function F, which provides allowable probabilities of the non-authorized access (P_{naa}), integrity violation (P_{iv}) and service denial (P_{sd}) at given temporal ($t_F \leq t_a$) and resource constraints ($R_F \leq R_a$):

$$1 - \prod_{i=0}^{T} \left[P\left(F(Func^1(t_i)) = N^1(t_i)\right)\right] < P_{naa}^{a},$$

$$1-\prod_{i=0}^{T}\left[P\left(F(Func^2(t_i))=N^2(t_i)\right)\right] < P_{iv}^{a},$$

$$1-\prod_{i=0}^{T}\left[P\left(F(Func^3(t_i))=N^3(t_i)\right)\right] < P_{sd}^{a},$$

where i - is the variable of time counting ($i=0,\ldots,T$), $P\left(F(Func^1(t_i))=N^1(t_i)\right)$, $P\left(F(Func^2(t_i))=N^2(t_i)\right)$, $P\left(F(Func^3(t_i))=N^3(t_i)\right)$ - are the probabilities of events occurring during the performance of the information system protected by ISS, at the moment of time t_i the messaging is carried out on the authorized access channels, the integrity violations and the service denial are absent. P_{naa}^{a}, P_{iv}^{a}, P_{sd}^{a} - are allowable probabilities of the non-authorized access, integrity violation and service denial respectively. t_F - is the common execution time of security functions. t_a, R_F , R_a - are anytime-constraints on providing the security functions (t_a - is the vector of temporary constraints on the performance of the security functions. R_F - is the vector of resources needed on security maintenance. R_a - is the vector of resources allowed to be used for security maintenance).

Let us represent a security function as a set of individual functions:

$$F=F_1\cup F_2\cup F_3\cup F_4\cup F_5\cup F_6\cup F_7\cup F_8,$$

where F_1 - are the functions of access control agents, F_2 - are the functions which are carried out by the audit and detecting non-authorized access agents (intrusion detection agents), F_3 - are the functions provided by means of anti-intrusion agents, F_4 - are the functions realized by the agents of assessing the damage of non-authorized access and information integrity recovery, F_5 - are the functions of cryptographic defense agents, F_6 - are the functions of steganography agents, F_7 - are the functions of authentication agents and F_8 - are the functions of meta-agents responsibility associated with management and coordination of individual agents activity to solve the general information defense task.

Let us assume that in the initial state there are standard (correct) copies of all software components of a protected information system and the standard configuration of hardware is created. At initial loading and also at restoration of damaged information, the. *authentication agents* authorize loading only of reference copies of the program components. User's input in the system is authorized, if he is registered in the system with the appropriate rights *(ID - password)*. For each user, the list of resources are set (which can be given to him/her according to his/her authority) as well as order (or priority) of their granting is determined.

The resources concession checking is carried out by the *access control agents* by realization of discretionary access control rules specifying to each pair "subject - object" the authorized kinds of messages (reading, recording, performance, etc.). The various access control agents cooperate to each other with the purpose of maintenance of non-discrepancy of discretionary access control rules on various sites of computer network. The information flows of various confidentiality are supervised by the access control agents by realization of mandatory access control rules not admitting outflow of confidential information.

The *authentication agents* carry out maintenance of conformity between functional processes realized and subjects (users, programs) initiated by these processes. While receiving a message from a functional process, the authentication agents determine the identifier of the subject for this process and transfer it to access control agents for realization of discretionary access control rules.

The *cryptographic and steganography agents* carry out creation of safe channels of an exchange between the computer network sites. These channels are multiplexed by the authentication agents and afforded to functional processes.

All actions of the subjects and objects are registered by the *audit and detecting non-authorized access agents (intrusion detection agents)*. As a result of statistical processing of the messages formed in the information system the intrusion detection agents can stop information processes, inform the security manager, and specify the discretionary access control rules. The statistical process is used for learning. These agents use the available information about normal functioning processes, possible anomalies, non-authorized access channels and probable scripts of attack.

The intrusion detection agents cooperate with the *anti-intrusion agents*, and also with the *agents of accessing the damage of non-authorized access and information integrity recovery*. The intrusion detection agents can generate the reports on behavior of information system subjects and transfer them to the security manager.

The host-based *meta-agents* carry out a management of information security processes. This includes coordination of actions of other agents and resolution of conflicts between them.

3 Ontology of Information Security Domain

In the multi-agent ISS, to solve the global task of information security in a distributed and cooperative fashion, agents must communicate to each other by message exchanges. Message exchange surmises that agents are able in some sense to "understand" each other. Mutual agent understanding means that (1) each agent "knows" what kind of task it must and is able to execute, (2) what agent(s) it has to address for its request for help if its functionality and/or information are not enough to deal with a problem within its scope of responsibility, and (3) agent's messages must be represented in a form and in terms that are understood to addressee. Therefore, each agent must possess its own model and models of many other agents of the system.

One of the most promising approaches to model the distributed agents' knowledge, beliefs and common ground of multi-agent-system as a whole, is the utilization of domain *ontology* [9-11]. This naturally follows from the motivation of using ontology for the purpose of enabling knowledge sharing and reuse. It is well known that the ontology-based approach can be applied to generate a consistent distributed knowledge base in many applications and for agent-based ISS as well.

Like any other domain, ontology of the information security domain is a description of the partially ordered concepts of this domain and the relationships over them that should be used by the agents. This ontology describes, in a natural way, *ontological commitments* (constraints, social rules, and etiquette) for a set of agents so that they might be able to communicate about a domain of discourse without necessarily operating on a globally shared theory. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms [10].

A part of the developed fragments of the information security domain ontology is depicted in Fig.2 through Fig.5. It is a part of the ontology that is associated with tasks of access control agents, the agents for audit and detecting non-authorized access, the authentication agents and finally, the meta-agents.

The *access control agents* carry out two basic functions [17]: (1) to operate flows of information with various degrees of confidentiality, not allowing the outflow of sensitive information on non-authorized access channels; (2) to provide access of users to information resources in strict conformity with their functional role. The first function is implemented by means of performance of mandatory access control rules, and the second function by means of discretionary access control rules. Therefore, as a basis of the construction of access control agents, the models of implementation of mandatory and discretionary access control rules should be fixed. The model of realization of the mandatory access control rules establishes conformity between the actual degrees of information confidentiality and their internal performance in information systems. This model determines the procedures of realization of mandatory access control rules. The model of realization of discretionary access control rules should allow one to define the concrete subjects and objects the discretionary access control rules, and to deduce one rule from others as well as to set conformity between access control rules of various depth.

The *audit and detecting non-authorized access agents (intrusion detection agents)* are intended for identification of the non-authorized intrusion into information systems from the outside, definition of deviations of registered users' actions from the prescribed order, which consequence can become the non-authorized access to information, and also search of the not documentary functions and mistakes in hardware and software. These agents should carry out all audit functions and furthermore, execute statistical data processing measuring the behavior of the subjects in relation to

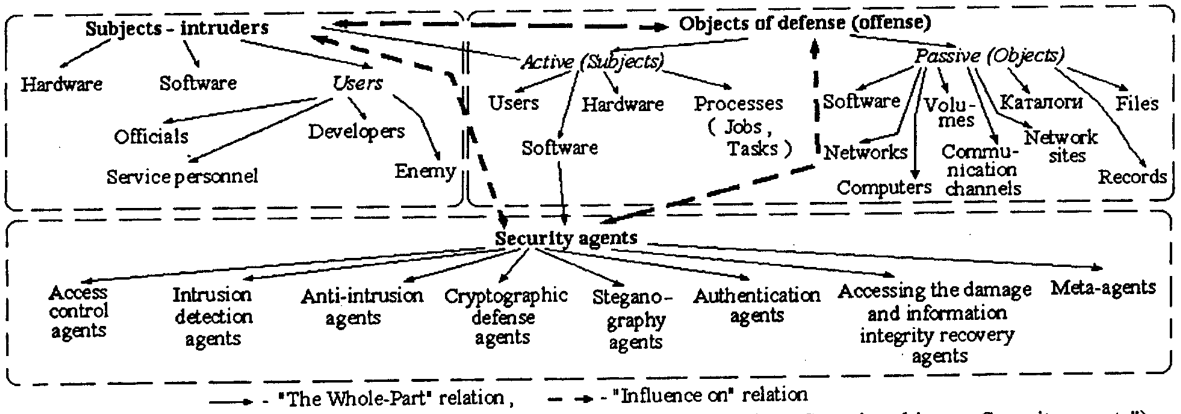
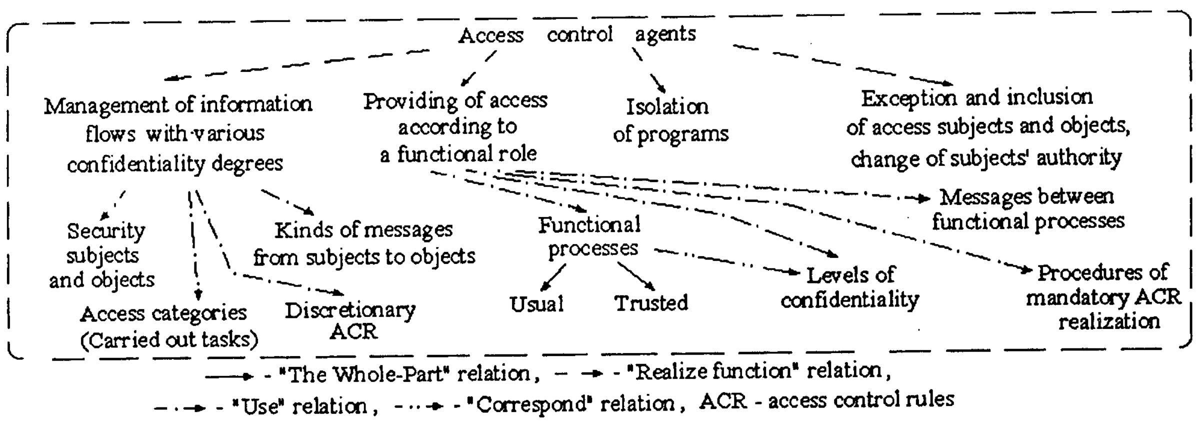


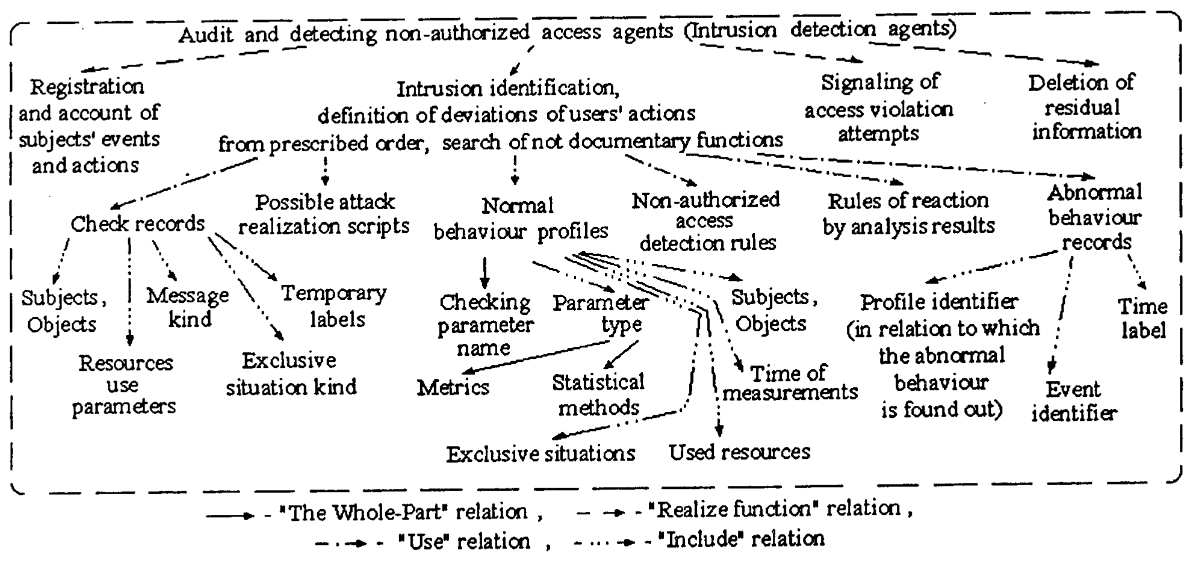
Fig.2. Fragment showing ontology of information security domain ("Intruders- Security objects - Security agents")

various objects. This is done on the basis of deviations from normal behavior as well as attributes characterizing non-authorized actions. The necessity of intrusion detection agents is precipitated by opportunities of intrusion into information systems from the outside. Furthermore, the unknown probability of *backdoors* or undocumented function calls , created casually or deliberate, can be very high. The models of the non-authorized access detection realized by these agents describe a process of data gathering about behavior of the users and all functional component behavior of the users and all functional components of the information system. The agents form required data structures, determine methods of statistical processing for reception of the characteristics of normal and abnormal behavior as well as the actions (which are undertaken during deviations from normal behavior). With the purpose to increase the efficiency of the non-authorized access detection, these agents should apply non-authorized access detection rules, which allow one to detail the access control rules, to conduct search and elimination of non-authorized access channels in real time.



Access control agents
Management of information flows with various confidentiality degrees
Providing of access according to a functional role
Isolation of programs
Exception and inclusion of access subjects and objects, change of subjects' authority
Messages between functional processes
Security subjects and objects
Kinds of messages from subjects to objects
Functional processes
Levels of confidentiality
Procedures of mandatory ACR realization
Access categories (Carried out tasks)
Discretionary ACR
Usual
Trusted
- "The Whole-Part" relation, - "Realize function" relation,
- "Use" relation, - "Correspond" relation, ACR - access control rules

Fig.3. Fragment of the ontology of information security domain ("Access control agents")



Audit and detecting non-authorized access agents (Intrusion detection agents)
Registration and account of subjects' events and actions
Intrusion identification, definition of deviations of users' actions from prescribed order, search of not documentary functions
Signaling of access violation attempts
Deletion of residual information
Check records
Possible attack realization scripts
Normal behaviour profiles
Non-authorized access detection rules
Rules of reaction by analysis results
Abnormal behaviour records
Subjects, Objects
Message kind
Temporary labels
Resources use parameters
Exclusive situation kind
Checking parameter name
Parameter type
Subjects, Objects
Profile identifier (in relation to which the abnormal behaviour is found out)
Time label
Metrics
Statistical methods
Time of measurements
Event identifier
Exclusive situations
Used resources
- "The Whole-Part" relation , - "Realize function" relation ,
- "Use" relation , - "Include" relation

Fig.4. Fragment of the ontology of information security domain ("Audit and detecting non-authorized access agents")

The *authentication agents* goal is to distribute safe channels of message exchange between information processes, to sanctify the processes initialization only to given set of the users both information system components and to maintain documentary confirmation of information exchange. The necessity of the authentication agents is caused by the following reasons: (1) there is a need to exempt the discretionary access differentiation agents from additional calculations connected to authorize transfer and change of subjects' roles; (2) while conducting distributed information processing, the message made by some user causes a set of parallel flows of the messages processable on various information processing sites. Thus, for realization of discretionary access through differentiation rules on each site, it is necessary to give to access control agents the identifiers of the subjects responsible for message generation; (3) the messages belonging to one processes, can be intercepted and changed by other processes, therefore it is necessary to give to processes the channels providing integrity and confidentiality of transmitted messages.

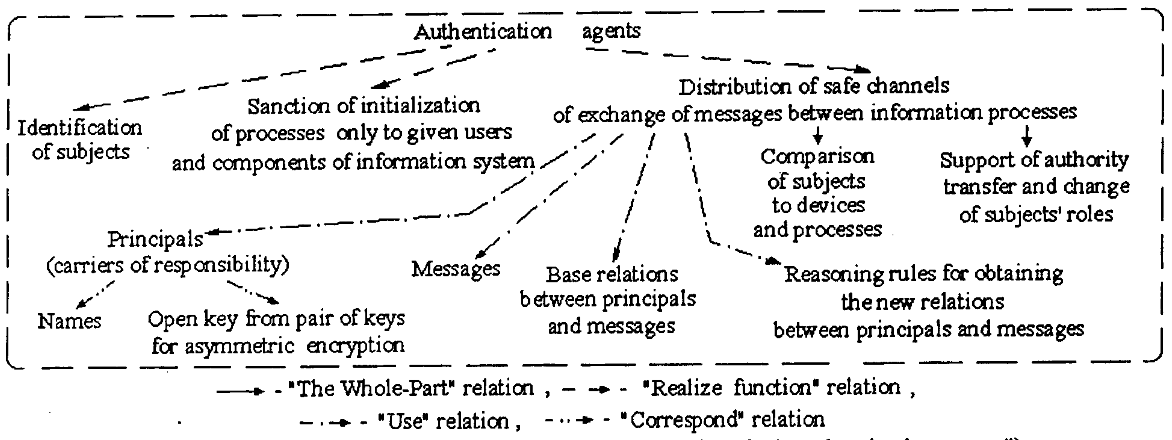

Authentication agents
Identification of subjects
Sanction of initialization of processes only to given users and components of information system
Distribution of safe channels of exchange of messages between information processes
Comparison of subjects to devices and processes
Support of authority transfer and change of subjects' roles
Principals (carriers of responsibility)
Messages
Base relations between principals and messages
Reasoning rules for obtaining the new relations between principals and messages
Names
Open key from pair of keys for asymmetric encryption
- "The Whole-Part" relation , - "Realize function" relation ,
- "Use" relation , - "Correspond" relation

Fig.5. Fragment of the ontology of information security domain ("Authentication agents")

The authentication models constituted on the basis of the authentication agents' functioning describe the order of granting of the safe channels to information processes. They also describe the order of determination of the subjects, to which discretionary access differentiation rules can be applied. These models specify a transformation from logic descriptions to the certificates which identify against a fake and are the basis of users' authentication during the input of the information system and during loading and initialization of software components. The given models allow one to provide documentary confirmation of transfer and reception of messages by processes, and also transfer the delegation of authority of one subject or another.

It is necessary to emphasize the important role of *meta-agents* for coordination of security agents' actions. Let us explain the role of meta-agents in supporting of interaction between the access control agents and intrusion detection agents. The performance of security functions by access control agents has the following features: (1) message checking is carried out in consecutive order with functional operations of the information system (because the message check on conformity to certain access control rules will not be carried out, functional operation performance is impossible), that reduces productivity of information system; (2) increase the access control depth which causes an avalanche increase of quantity of the messages which are necessary for supervising which essentially has an effect on productivity; (3) an increase of access control depth is limited to opportunities of formation of access control rules, because the set of details of the hardware and software performance appears latent from the developers; (4) insufficient depth of access control does not allow one to check messages of bottom levels, and therefore, leaves an opportunity to detour the access control rules; (5) message generation on a non-authorized access channel is a relatively rare event, therefore checking of the most part of messages is carried out during idle CPU periods.

The audit and detection of non-authorized access agents carry out their processes basically in parallel with functional processes, interrupting them only at detection of non-authorized access.

Meta-agents solve management and coordination of decisions of the subordinate security agents' during their performance. They should (1) ensure flexibility of distribution of functions between access control and non-authorized access detection agents, (2) optimize the time required for protection of information system performance on intervals belonging to the critical path while providing an admissible level of probability of non-authorized access as well as minimizing resources needed for detection of non-authorized access .

4 Agent-based Architecture of ISS

Let us consider a number of basic principles of construction of ISS as an integrated agents' community distributed in a network environment and allocated on several hosts (see Fig.6, Fig.7).

Let us suppose that each security agent should be host-based and run on some segment of the computer network (Fig.6). In this case, we assume a meta-agent is host-based as well. This meta-agent manages a set of the above mentioned specialized agents, which, in their turn, receive information from the agents - "demons" investigating the input traffic (login, password, IP address of source of input message, contents of query, etc.). The agents - demons carry out monitoring of the input traffic to different servers located on the same host, i.e. www-server, ftp-server, telnet-server, etc. In essence, they are software sensors that form various metrics of input traffic. They may be built in the way like considered in [4] and, for this reason, specific details are not shown below.

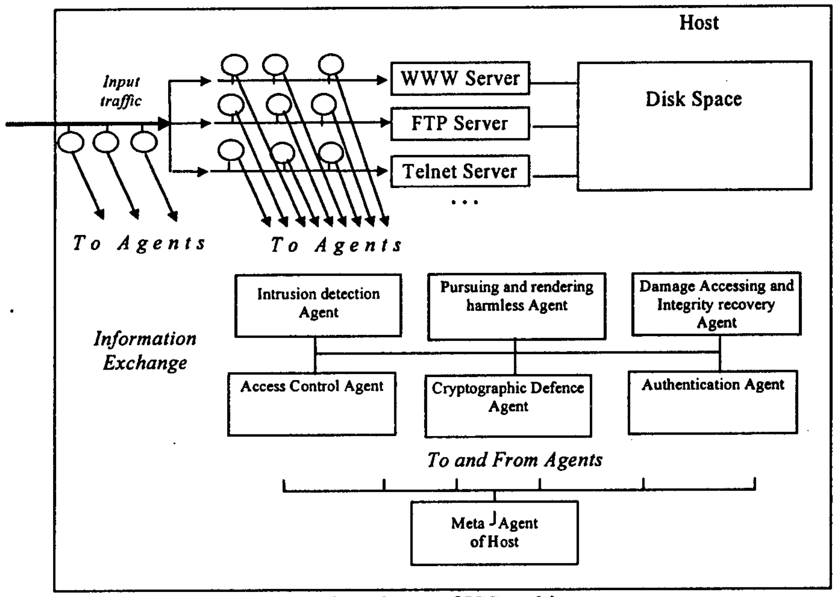


Fig.6. Host-based part of ISS architecture

All agents are permitted the possibility to communicate each other to provide the following ISS properties (fig.7):

(1) ISS has to be capable to detect network attacks, even if intrusions are undertaken "locally" during a given time interval;

(2) ISS should be capable to detect network attacks when attempts of the non-authorized intrusion are undertaken serially (for example, the substitution of a log-file, entrance with incorrect id/password), each of which separately can not be interpreted as attempt of intrusion. Together they represent an attack.

The offered set of agents can be on any host and can cooperate through the manager - meta-agent, which operates with the "top level" knowledge base and makes conclusions within the framework of one host.

The information interchange between hosts is carried out either on a peer-basis, or by means of the meta-agent which is the network layer manager.

In the first case, meta-agents of hosts cooperate among themselves. The initiative and conducting role is played by the first meta-agent, which has suspected intrusion. This agent receives information on its inquiries from the meta-agents of other hosts.

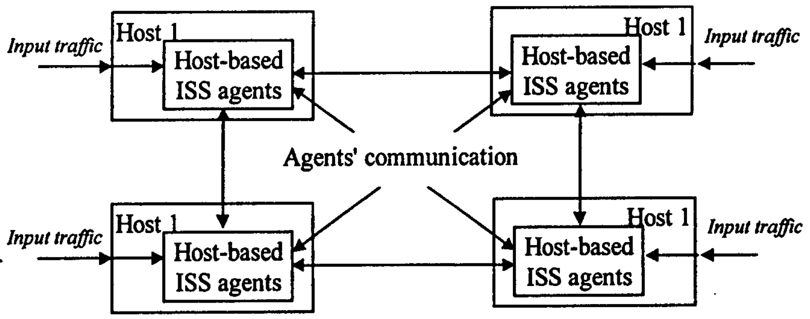


Fig.7. Network-based representation of ISS architecture

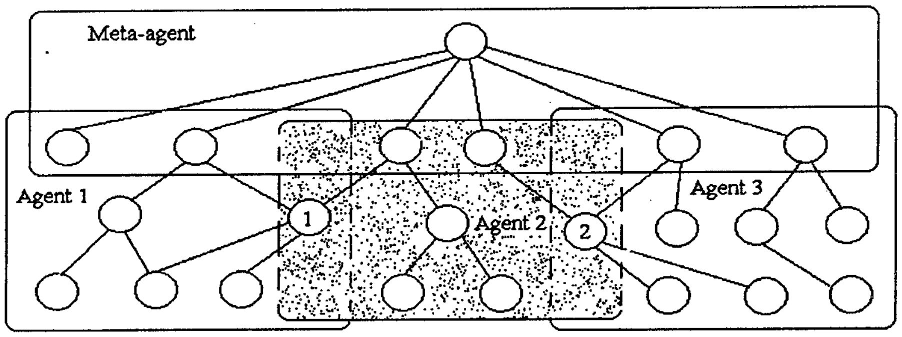


Fig.8. Abstract representation of agents' ontologies intersection

5 Agents' Coordination and Negotiation

The agents' functions determine those subsets of nodes and relations of ontology, which should be used by agents for the task solving. The nodes, placed on crossings of ontology fragments set by functions of separate agents, form knowledge, which is used in pairs of agents (see Fig.8). These nodes form the field of knowledge which is common for them for the acceptance of decisions. For acceptance of decision by agent 2 this agent needs to know something about nodes 1 and 2. But the agents 1 and 3 have more detailed knowledge of these nodes. The agent 2 should receive this knowledge from them. Similar situations will take place for other agents. The agent 2 knows "only" about nodes 1 and 2, but it can set the question in the terms understandable for agents 1 and 3, which can receive from them knowledge and can this knowledge be correctly interpreted.

In Fig.9 the example of common fields of knowledge for the meta-agent, access control agent and authentication agent are shown. The meta-agent knows what functions the specialized agents realize, and addresses to a specific agent for realization of these functions. The access control agent needs to know whether the given access subject has the certain authority. For this purpose it addresses to the authentication agent receiving from it the appropriate information.

Let us consider an example of script of security agents' interaction within the framework of the offered architecture of ISS. Let the protected information system include three hosts which are distributed in a network (see Fig.10).

1. The destroying software (for example, a new combined file virus) penetrates into the information system on the host 1, avoiding the control access agents. The attempts of penetration are made also on other hosts of information systems. In host 1, the virus carries out a number destruction actions (for example, infects executable files, reads out passwords, tries to transfer this information on a network, damages files of documents).
2. One of the intrusion detection agents situated on the server 1 reveals (by means of abnormal actions and their consequences) a virus presence and transfers to the meta-agent the attributes of virus actions (for example, increase of exe-files to 1636 bytes, data about attempts to leave for an external network) and also information about consequences of its actions (modified files of documents and reference to password files).
3. The meta-agent (on intrusion) notifies the system manager and meta-agents of other hosts, and also calls for the anti-intrusion agent, transferring the appropriate information on the intrusion.
4. The anti-intrusion agent identifies and neutralizes a virus, and then transfers to the meta-agent the information about the carried out actions and specified virus parameters (including information on sending message address).
5. The meta-agent causes the agent of accessing the damage of non-authorized access and information integrity recovery, and also transfers the information about a virus to meta-agents of other sites of information system (hosts 2 and 3).
6. The agent responsible for accessing the damage of non-authorized access and information integrity recovery specifies consequences of virus actions and restores the damaged files.

6. Related works

Many of the existing and proposed ISSs use a monolithic architecture. Several approaches that exploit the idea of distributed ISS are given in [15, GrIDS project], [12, the NADIR system], [18, Cooperative Security Manager], [14, EMERALD project]. In the paper [8] the metaphor of immune system is exploited to develop a widely distributed program to intrusion detection problem solving. There exist few papers, for example, [1, 4, 5, 12, 13, 18] that consider an agent-based approach for an information security system design. All these papers consider only a separate task of the information security, in particular, the so-called intrusion detection task. It must be noticed that all of them use a very simplified point of view on the agent itself, their architecture, and functionality and on agents cooperation. Thus, in [1, 4, 5] so-called

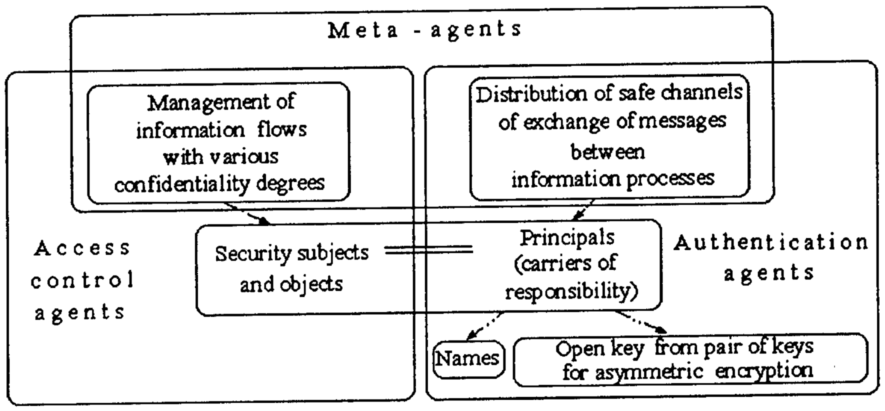

Meta - agents
Management of information flows with various confidentiality degrees
Distribution of safe channels of exchange of messages between information processes
Access control agents
Security subjects and objects
Principals (carriers of responsibility)
Authentication agents
Names
Open key from pair of keys for asymmetric encryption

Fig.9. An Example of agents' ontology intersection

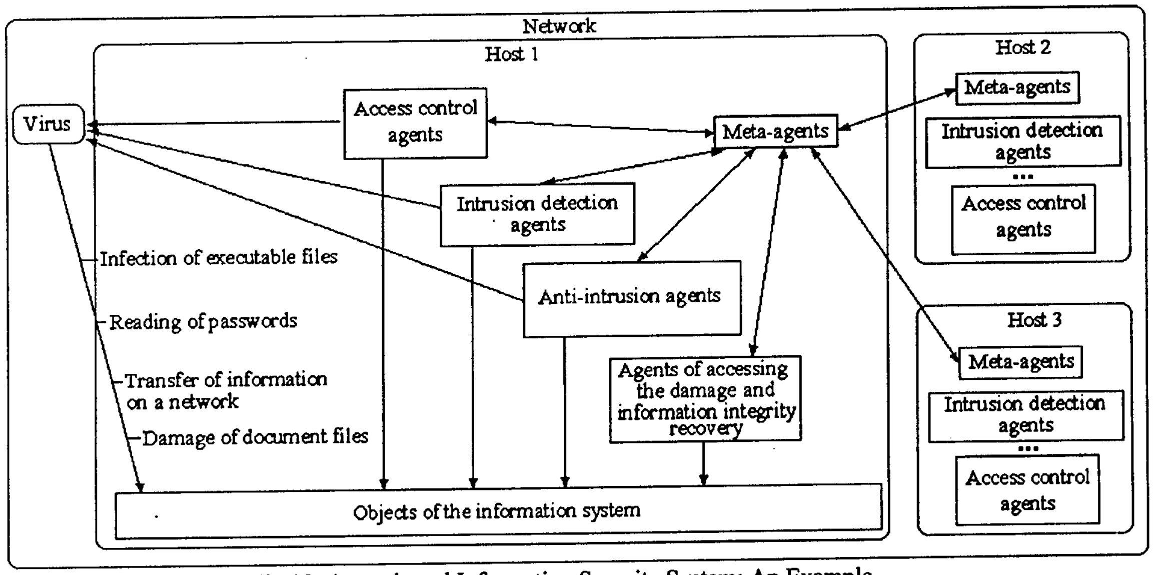

Network
Host 1
Host 2
Host 3
Virus
Access control agents
Meta-agents
Intrusion detection agents
Anti-intrusion agents
Agents of accessing the damage and information integrity recovery
Infection of executable files
Reading of passwords
Transfer of information on a network
Damage of document files
Objects of the information system
Meta-agents
Intrusion detection agents
Access control agents
Meta-agents
Intrusion detection agents
Access control agents

Fig.10. Agent-based Information Security System: An Example

AAFID architecture of intrusion detection system is proposed. It utilizes the notion of an agent that mainly coincides the traditionally used notion of low level "demon", i.e. a program that is attached, say, to a port "to inspect the content of network packets and to perform operations based on this information" [4]. AAFID ISS itself can be distributed over a number of hosts in a network and may contain a great number of such agents. Each of them monitors a small aspect of entire network traffic to recognize in a sense "a probably suspicious behavior", say, not known IP address of input packet, an attempt to write information on a hard disk, etc. Each agent is "measuring" an attribute of input traffic or something else and comparing its value to an assigned "threshold". A "suspicious behavior" corresponds to the case when measured value overcomes the above threshold. Agents cooperate together via sending information to the so-called transceiver that is host-based aiming at detection of an intrusion on the basis of entire amount of information obtained by a host-based agency as a whole. An agent may also perform a simple function (say, a linear threshold function of input variables) which arguments are outputs of a group of agents. A hierarchical structure (host-based and network-based agents $\rightarrow$ host-based transceivers $\rightarrow$ network-based monitor) is imposed over a multi-agent system. Since agents may be located on or migrate to different hosts, the approach utilizes advantages of the host-based and the network-based ISS. Unfortunately, (1) other authors restrict themselves by solving only one of the information security related tasks, i.e. intrusion detection task, (2) do not pay needed attention to the agent cooperation problem and multi-agent system architecture because the main goal of the paper is development of training procedures ("learning by feedback") via Genetic Programming that makes possible to assign to each agent the optimal value of the above mentioned threshold or linear threshold function. In addition, (3) these papers ignore advantages of using intelligent agents. Nevertheless, even using such a relatively simple agent-based approach as a model of ISS leads to a number of advantages such as efficiency, fault tolerance, resilience to subversion, scalability, trainability, etc.

In our approach we have borrowed the idea of using low level agents to analyze input traffic and other data sources but we consider them only as sensors for an intelligent multi-agent system.

A multi-agent system for intrusion detection is considered in the paper [18] which a Cooperating Security Manager (CSM) is proposed. It is closer to the modern understanding of multi-agent system. CSM runs on each computer connected to a network and aimed at cooperative detection of probable intrusion. Its architecture include sensors that analyze users activity and input queries to the system to recognize abnormal system usage patterns. The entire system contain a number of host-based sensors that cooperate via information exchange that, in its turn, makes it possible to detect attacks in a host as well as in the network as a whole. For example, several agents based on different hosts detect the sequential attempts of entry having incorrect login and password and all of them have the same IP address of source than no one agent separately is able to detect an attack but, to be analyzed as a whole, this information is the certain feature of attack on the network. It is a unique way for agent cooperation. Unfortunately, like previous approaches, the last one is based on a relatively poor agents functionality, architecture and the way of cooperation. In addition the approach is aimed at solving the only information defense task, i.e. intrusion detection task. Nevertheless, even such a relatively simple approach demonstrates a number of promising advantages of an agent-based model of ISS to detect network-based attacks.

7. Conclusion and Future Work

In this paper an agent-based model of Information security system is proposed based on ontology (a network with a sense of existence). The main features of the approach are as follows:

• An extendible information security system is considered as a number of host-based and network-based specialized agents and managing intelligent agents that solve, jointly, the entire multitude of tasks of information security, such as (1) access control, (2) detection of non-authorized access, (3) authentication, (4) anti-intrusion task, (5) assessing the damage of non-authorized access and information integrity recovery, (6) cryptographic defense and (7) steganography and steganoanalysis.

• Agent cooperation and message interpretation is based on the use of *ontology of information* security domain. The latter is considered as the framework for distributed common knowledge and agent's individual knowledge development and representation. In addition, use of ontology in such a way forms the model of common ground needed to reach an agents mutual understanding during the message passing process.

The main paper results include:

(1) development of information security domain ontology that is associated with the multitude of information security tasks under consideration;

(2) development of an agent-based architecture of information security system that aims at solving the entire multitude of problems related to tasks;

(3) definition of agents' cooperation framework.

In future works it is planned to develop in more detail the domain ontology, the agent-based architecture and the formal frameworks for agent cooperation, and distributed knowledge and beliefs representation.

One more intention is to exploit "learning by feedback" methods to provide ISS by real-time adaptation properties. The latter is conditioned by the necessity for ISS to specialize to new kinds of network-based attacks, to variability of computer network structure and platforms, etc. In its nature, the tasks of providing information security cannot be formulated in any constant form. This is a reason of high importance of providing ISS by powerful dynamic and adaptable learning abilities.

A software prototype development to verify and validate the main ideas proposed in this paper and those that will be developed is being pursued for future work.

Bibliography

1. J.Balasubramaniyan, J.Garcia-Fernandez, D.Isakoff, E.Spafford, D.Zamboni. An Architecture for Intrusion Detection using Autonomous Agents. In Proceedings of the 14th Annual Computer Security Applications Conference. Phoenix, Arizona. December 7-11, 1998.
2. W.Brenner., R.Zarnekow., Wittig H. Intelligent Software Agents. Foundations and Applications. Springer-Verlag, 1998.
3. Canadian Trusted Computer Product Evaluation Criteria. Canadian System Security Centre Communication Security Establishment, Government of Canada. Version 3.0e. January 1993.

4. M.Crosbie, E.Spafford. Active Defending of a Computer System using Autonomous Agents. Technical Report No. 95-008. COAST Group, Purdue University, 1995, pp.1-15.
5. M.Crosbie, E.Spafford. Defending a computer system using autonomous agents. In Proceedings of the 18th National Information Systems Security Conference, 1995.
6. Common Criteria for Information Technology Security Evaluation. National Institute of Standards and Technology & National Security Agency (USA), Communication Security Establishment (Canada), UK IT Security and Certifica tion Scheme (United Kingdom), Bundesamt fur Sichereit in der Informationstechnik (Germany), Service Central de la Securite des Systemes (France), National Communications Security Agency (Netherlands). Version 1.031.01.96.
7. Federal Criteria for Information Technology Security. National Institute of Standards and Technology & National Security Agency. Version 1.0, December 1992.
8. S.Forrest, S.A.Hofmeyer, A.Somayaji. Computer Immunology. Communication of the ACM, vol.40, No.10, October 1997, pp.88-96.
9. T.R.Gruber. Toward principles for the design of ontologies used for knowledge sharing. In Proceedings of International Workshop on Formal Ontology, March 1993. Stanford Knowledge Systems Laboratory Report KSL-93-04
10. T.R.Gruber. What is an Ontology? http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.
11. Guarino. Formal ontology, conceptual analysis and knowledge representation. Int. J. Human-Computer Studies, No.43, 1995, pp.625-640.
12. Hochberg et al. "NADIR": An Automated System for Detecting Network Intrusion and Misuse. Computers and Security, vol.12, No.3, 1993, pp.235-248.
13. T.Lunt et al. Knowledge-based Intrusion Detection. In Proceedings of 1989 Governmental Conference Artificial Intelligence Systems. March, 1989.
14. P.A.Porras, P.G.Neumann. EMERALD: Event monitoring enabling responses to autonomous live disturbance. In Proceedings of 20-th National Information System Security Conference. National Institute of Standards and Technologies, 1997.
15. S.Stainford-Chen, S.Cheung, R.Crawford, M.Dilger, J.Frank, J.Hoagland, K.Levit, C.Wee, R.Yip, D.Zerkle. GrIDS: A Graph-based Intrusion Detection System for Large Networks. In Proceedings of the 19-th National Information System Security Conference. Vol.1, National Institute of Standards and Technology, October, 1996, pp.361-370.
16. Trusted Computer System Evaluation Criteria. US Department of Defense 5200.28-STD, 1993.
17. D.I.Voitovich, I.V.Kotenko Protection of distributed computing systems against the non-authorized access to information. Reports of 49-th Scientific Session devoted to Day of radio. Moscow. 1994. pp.66-67 (in Russian).
18. G.White, E.Fish, U.Pooch. Cooperating Security Managers: A Peer-Based Intrusion Detection System. IEEE Network, January/February 1996, pp.20-23.

INTELLIGENT PROCESSING OF WEB-RESOURCES: ONTOLOGY-BASED APPROACH AND MULTIAGENT SUPPORT

Vladimir F. Khoroshevsky [1], Natalia V. Maikevich [2]

[1] *Artificial Intelligence Problems Division, Computer Center, Russian Academy of Sciences, 117967, 40 Vavilov str., Moscow, RUSSIA,*
E-mail: khor@ccas.ru
[2] *Artificial Intelligent Centre, Program System Institute, Russian Academy of Science, 152140, Pereslavl-Zalessky, RUSSIA,*
E-mail: nut@botik.ru

Abstract
The representation of the information as knowledge for intelligent processing of Web resources is the theme of the paper. Appropriate models of ontology and ontological system for the Web-oriented knowledge instead data spaces representation is discussed. The project of multi - agent system for ontology-based information retrieval on Web is presented.

Keywords: *Ontology, knowledge representation and processing, knowledge space, intelligent information retrieval, multi - agent systems, WWW.*

1. Introduction

It can be depicted that Web intellectualisation nowadays is connected with the using of explicit knowledge representation methodologies, methods and tools developed in AI-community [3, 4]. There are many approaches devoted to the representation of semantics for Web resources, that may be viewed as attempts to the creating of knowledge spaces instead of data spaces on Internet [13]. Among others taxonomies, large databases and ontologies are well known [17]. And the notion of ontology is the key one in all of them. The main questions in the domain nowadays are the following: what ontologies should be, where their place within the intelligent Internet-oriented systems, how to design and develop the ontologies for such a systems and what a technology is appropriate to their implementation.

The main goal of presented paper is the discussion of the ontologies and ontological systems models and an approach to the design and processing of ontology within intelligent Internet-oriented systems. The appropriate multi - agent system FireExpert for ontology-based information retrieval on Web is presented. Presentation below is organised as follows. Next section presents the background in the domain, overview of ontology-based projects, ontologies models and approaches to the processing of ontologies. Then a special formalism for the ontology description and processing is discussed. Project of multi - agent system FireExpert that is in progress nowadays in Russia is presented in the conclusive section of the paper. Last section summarises the results and depicts the directions of future researches.

2. Related Researches and Developments

There are some problems in the design and implementation of knowledge spaces on WEB. First of all, if we would like to create such a space within the Internet we should use an appropriate technology and tools for the networking. On the other hand if we would like to receive really intellectual space we need in explicit representation of its model. So we need in a technique and tools for knowledge acquisition, representation and processing. And finally, if we would like to develop significant software system supporting our intellectual space we need in appropriate software engineering technology. So knowledge spaces design and development is based on amalgamation of methods and tools from

- WWW-technology
- KBS-paradigm
- MAS-approach

rested at the object-oriented implementation methods and tools.

According to depicted problems and overview of the results receiving in background domains [3, 5, 8, 9; 16; 17; 22;] it is possible to outline that we can actively use:

- from the WWW-technology networking in open environment and browsing,
- from KBS-paradigm - explicit knowledge representation and semiotic approach to knowledge processing
- from MAS - ideas of agents working in asynchronous manner and multilevel architecture.

As it was mentioned above the core of presentation is discussion of the ontology-based paradigm implementation to the knowledge spaces design and development and using of multi - agent support for the implementation of such systems. So we concentrate below at the KBS-paradigm and MAS-approach.

2.1. What Ontologies Projects are "living" on Web

There are number of projects somewhat connected with ontologies on Web today. See for example the overview [8]. Below we will enlarge on the several most interesting (from the paper point of view) works.

It seems that there are two classes of the projects in domain nowadays: with hot spot on ontology itself and connected with ontologies support development methodologies and tools.

The analysis of the "ontology oriented" projects has indicated that ontologies themselves can be classified in according to the following several main dimensions:

- dependency from the task and/or domain;
- ontology axiomatic level;
- domain nature (application - metalevel).

For example, the main goal of CYC project [15] is creating an ontology for common-sense knowledge and appropriate inference engine. The CYC's knowledge base contains about 10,000 terms with assertions about each. The representation language CycL and inference engine that performs general logical deduction are developed in the project.

Top level ontology oriented to the natural language (English, German and Italian) processing is developed within the Generailized Upper Model (GUM) system [29]. This ontology level is between lexical and conceptual. GUM ontology model is based on taxonomy only organising as concepts hierarchy and standalone links hierarchy.
TOVE (Toronto Virtual Enterprise) [21] is an example of the domain-oriented ontology. The appropriate system developing within the project can respond to user's questions (using explicitly representing knowledge and/or deduce answers) about enterprise modelling. Formal level of TOVE ontology is specified by frames. There isn't facilities to the integration of the ontology with another ontologies that "live" outside of TOVE.

Gensim [30] is an another example of the domain-oriented (biochemical reactions modelling) ontology. It consists of two parts: objects (for example, molecules, proteins, etc.) ontology and description of objects potential behaviour. Frame formalism is used to the ontologies specification in Gensim project.

Another approach to ontology specification is used in Plinius system [31] oriented on semiautomatic data mining from texts. Atomic concept sets and complex concepts generation rules are defined in Plinius instead of explicit notions taxonomy. Frame formalism is used to the specification all ontology components.

So there exists a number of ontologies from such global as the CYC to various, more domain and task restricted ontologies such as, for example, TOVE and Gensim. As it was mentioned in [4] there are no global differences between knowledge representation for global and specific ontologies. Main disadvantage of large ontology is connected with the fact that such ontologies composed from abstract point of view at the domain subjects and their relationships. And to build single ontology reflecting the different views is not easy because it requires co-ordination between different people on different aspects [1]. In fact for each user exists own context that is dictated by the situation and the user's world model.

Instrumental projects in domain are concentrated (first at all) on the following problems:

- ontology design;
- sharable and reuse ontologies and
- composing different ontologies.

For example, SHOE (Simple HTML Ontological Extensions) project [17] presents the Internet-oriented information retrieval system using special ontologies and knowledge based inference. First at all SHOE tools are oriented on

- ontologies creation and
- HTML pages annotation.

SHOE ontologies specifications include classes and/or categories ISA-hierarchy, atomic relationships between categories and a set of Horn like inference rules. There is no central provider of ontology in the system developing within SHOE project. As a consequence of this fact user can't know about some new concepts and doesn't use them in his requests, then answer may not contain important information user needed.

It is supposed that special tags expanding HTML specify ontologies in SHOE project. The same approach is used here for HTML-documents semantically annotation by their owners [17]. The authors can expand basic ontology with new concepts, inference and classification rules.

Ontobroker project [7] is close to SHOE by its goals. But main idea of Ontobroker system is using "newsgroup" metaphor to define a set of people having common view on ontology subjects, e.g. common model of world. Powerful support tools develop within the project for knowledge acquisition from Web-documents.

There are three main modules in Ontobroker: queries forming text and graphical interfaces, inference engine and special Internet robot. Each of them is supported by special language: query, knowledge representation and annotation languages. To inference and to representation language for ontologies Ontobroker based on Frame Logic [14]. The special Ontocrawler search engine finds and collects all "onto-compatible" HTML-documents.

KA^2 (Knowledge Acquisition Initiative) is open initiative and participants take part in process of creating distributed ontology [1] to organise intellectual knowledge acquisition from Web documents. Ontology plays role of the basis to annotate HTML-documents in the project and Ontobroker is the basis for the solving the last task.

The analysis of above projects shows that main time and labour consumption tasks in the ontologies based systems using are ontologies specification and Web-documents annotation. So there are some projects oriented on the design, development and support special ontological servers.

It is possible to outline Ontolingua Server project [6] in the domain. Main server's functions are in support of the ontologies libraries processing, creating new ontologies and editing already exist ontologies. Server can be used in the following regimes: remote users groups working (view, create, support and save ontologies from Web browser via HTTP protocol); working with ontologies from remote applications (modification, data queries, data updating via API data transmission protocol); ontologies converting (among others, CORBA's IDL and KIF are supported).

So the main tasks in ontologies creating are connected with sharable and reuse ontologies; ontology design; comparing and composing different ontologies [8]. The processes of ontologies design and process are the questions that are taking attention of ontology researchers today. Now almost all existing and newly appearing ontologies suppose the one scheme of theirs organisation, have internal explicit knowledge representation language and some of them also suppose an inference engine. And different approaches concentrate on the ways of defining and accessing of domain topics and their inter relationships.

2.2. What MAS Technology has and What We are Need in

Agent technologies are usually provide architecture families, agents types and their models, components libraries and support development tools for different types of MAS.

Main MAS architectures [26] widely used nowadays with their characteristics that are presented in Tab.1.

Table 1.

Architecture	Knowledge representation	World model	Problem solving
Deliberative	Symbolic	Formal theory	Inference in formal theory
Reactive	Non-symbolic	Heurist ic model	"situation" -"act" type inference
Hybrid	Mixed	Hybrid model	Mapping on tasks solving methods

There are many pro and contra for each of above depicted architectures. But it seems that the hybrid architectures are the hot spot in current researches [9, 11, 12].

It is known [23] that software agents are viewed as an autonomous software construction with capabilities of executing without user intervention. There are some agent classifications presented in publications nowadays [12, 26]. One of them is presented in Tab. 2.

Table 2.

Agent Types / Attributes	S	Sm	Int	Tr Int
Autonomous execution	+			
Communication with other agents or the user	+	+		
Monitoring of execution environment	+	+		
Ability to use symbols and abstractions		+	+	
Ability to exploit domain knowledge			+	
Capability of adaptive goal-oriented behaviour			+	
Learning from the environment			+	+
Tolerant to error, unexpected, or wrong input				+
Real-time execution				+
Natural language communication				+

Agenda:

S - simple agents	Int - intelligent agents
Sm - smart agents	TrInt - truly intelligent agents

The overview of the related papers in domain [25, 26] and analysis of the data presented in Tab. 2 show that the attributes of different agents types are not orthogonal. And more, different projects depict different attributes for the same types of agents [24]. But due to the goals of this paper it is possible to outline the following main attributes of the agents suitable for the processing of Web-resources:

- Autonomous execution;
- Communication with other agents or the user;
- Ability to use symbols and abstractions;
- Ability to exploit domain knowledge;
- Capability of adaptive goal-oriented behaviour;
- Learning from the environment and
- Natural language communication.

It seems that in context of the present paper the following pragmatically agents classification [23] is more useful:

- interface agents;
- information agents;
- intelligent agents;
- ontologies agents;
- agents of Internet-resources and
- broker agent.

So it will be used below as the basis but it is necessary to outline that above depicted in Tab. 2 attributes (all or some) should be implemented within each of pragmatic-oriented agent.

There exist some libraries and support development tools for the agents and MAS implementation nowadays. Well-known examples of appropriate software are presented on site [23]. Almost all of them are oriented to support of maximum smart agents implementation, and knowledge needed to the "living" of such agents are incorporated into program code by developer at the base of the intuition and experience.

We need in intelligent software to support the full-scale life cycle of MAS design, development and implementation. Attempts to create such tools are in progress now [11]. This is a global task that can be solved after the gathering of MAS design and development experience. That's why currently the local task is the development of MAS-specification formalisms based on knowledge processing methodology "caught" from AI. In fact we get in such case special intelligent toolkit based on appropriate expert system. And the composition of the traditional AI tools and MAS methods can give us the needed technology of MAS developing in Web environment.

3. Ontology and Ontological System Models and Remarks to Their Processing

In the frame of FIPA (Foundation for Intelligent Physical Agents) specifications of ontology [18] we may refer to it as to a particular system of categories accounting for a certain vision of the world. In its most prevalent use in AI an ontology refers to an engineering artefact constituted by a specific vocabulary used to describe a certain reality plus a set of explicit assumptions regarding the intended meaning of the vocabulary words usually in the form of a first-order logical theory. In the simplest case ontology is described by hierarchy of concepts and predefined relationships between concepts. In more sophisticated cases suitable axioms and inference rules are added in order to express other relationships between concepts and to constrain their intended interpretation. So ontology is the explicit expression of the conceptualisation. The latter concerns the way an agent structures its perceptions of the domain, while the former gives a meaning to the vocabulary used by the agent to transfer such a perceptions in the communication process. The ontology for the domain can be explicit constructed or implicit coded in appropriate agent's implementation [10].

According to the more concrete definition from the Stanford Ontolingua library [6], ontology is an explicit specification of some topic. From the practical point of view it means a formal and declarative representation of the topic including the vocabulary (or the list of constants-terms) for the referring of the domain terms, the integrity constraints on the terms, the logical statements that restricts the terms' interpretation and their relationships. Ontologies therefore provide an environment for representing and communicating knowledge about some topic.

There are distinguished:

- *top (meta)* ontologies describing very common concepts (such as space, time, event, act, etc.)
- *domain and task* ontologies connected with generic domain (for example, multi agent systems or entertainment) or with generic task or activity (for example, design & development or information retrieval) describing the domain oriented vocabulary with the description of the items specialisation from top ontology.

In presented paper we follow to the conception of the ontologies pairs - top and domain level based on the common formalism presented below.

3.1. Ontology Formal Model

Let's introduce the notion of an ontology formal model by the following manner:

Def. 1. An ontology formal model $\boldsymbol{O}$ is the ordered triple

$$\boldsymbol{O} = <X, \mathfrak{R}, \Phi>, \text{ where}$$

X - finite non-empty set of domain concepts and concepts' types;

$\mathfrak{R}$ - finite non-empty set of relationships between concepts;

Φ - finite set of interpretations (functions), defined on the concepts and/or relationships of the ontology $\boldsymbol{O}$ (axiomatisation).

It is clear that two components of ontology $\boldsymbol{O}$ (X and $\mathfrak{R}$) are not simple names but the complex structures that can be defined by the following way:

$[x \ \boldsymbol{is\_a} \ T_i \ O^{frame}; s_1 \ T_1^i \ y_1 \ O_1^{slot}; \ldots]$, where

x, s_m, y_m - domain variables and constants;

T_i, T_m^i - variables types;

O^{frame}, O_m^{slot} - special operations defined on the frames and their slots.

The partial order $\boldsymbol{is\_a}$ (":") is the main pre-definite relationship in model. All others have procedural defined semantics and implemented as demons.

To the specification of interpretations (functions) from Φ we use Java language with expanding it by the classes that are used nowadays in multi - agent community for the implementation of the agents.

So on the top-level view it is frame-based knowledge representation formalism. It's really but there are important distinguishes. The frames and slots composition is defined by the problem under solving (in our case ontology-based Web-resources processing). Some slots are interpreted as the relationships with dynamically defined and modified semantics; frame-oriented knowledge bases are interpreted as meta and domain specific ontologies connected with the different sides of the processes under design.

In according to above done definition and notation, the description of semantic networks structure used below for ontologies models representation can be the following:

```
[net is_a prototype;
    Nodes   {frame};Arcs    {frame}];
[node is_a prototype;
    FullName      string;
    ToArcs        {frame};
    FromArcs      {frame};
    BedirArcs     {frame}];
[arc is_a prototype;
    FullName      string;
    Source        frame;
    Destination   frame];
[Ontology is_a net;
    Inference   func, by_default
                SystemInferenceMachine () ];
[Concept is_a node;
    State    string,  one_of {"nonprocessed",
                              "active", "passive"},
             by_default "nonprocessed",
             if_changed Run (Interpretation);
    Links          {frame};
    URL            {string};
    Interpretation  func];

[Relationship is_a arc;
```

```
    State    string,  one_of {"nonprocessed",
                         "active", "passive"},
        by_default "nonprocessed",
        if_changed Run (Interpretation);
    Interpretation    func];
```

The processing of semantic networks is standard and can be described by the system inference engine with the following specification:

```
SystemInferenceMachine () {
  while ( ![$Target] ) {
    $TempAgenda = $Agenda;
    $WasActiveElem = false;
    while ( $TempAgenda != empty ) {
      $CurrElem = $Agenda>>;
      if ( [ $CurrElem : State] == "active") {
        Run ( [ $CurrElem : Interpretation] );
        $WasActiveElem = true;
      }else >>$TempAgenda = $CurrElem;
    };
    if ( $WasActiveElem == false ) {
      $Agenda = $TempAgenda;
      exit ("Target state is not succeed in the
              ontology");
    };
  };
  exit ("Target state is succeed in the
                               ontology:",
        [$Target]);
}
```

According specification the semantic network processing consists in waves activation. It is stopped when target state is succeeded.

It is clear that in such a manner can be solved diagnostic tasks (target state is the diagnosis, the set of initial states is the description of anemones, and the trace is the explanation of the decision). But the approach is not completely convenient for the intelligent processing of Web-resources, where we need in multilevel network representation of domain and more sophisticated inference mechanism.

That's why let's expand the ontology definition presented above onto the definition of an ontological system formal model:

<u>**Def. 2.**</u> An ontological system formal model Σ^O is the ordered triple

$$\Sigma^O = \langle O^{meta}, \{O^{d\&t}\}, \Xi^{inf} \rangle, \text{ where}$$

O^{meta} - top level ontology;

$\{O^{d\&t}\}$ - the set of domain ontologies and ontologies of tasks, and

Ξ^{inf} - the model of associated with an ontological system Σ^O inference machine.

There are several types of the problems can be solved within the ontological system model.

For example, if we suppose that

$$\{O^{d\&t}\} = \{ O^{EC}, O^{Search} \}, \text{ where}$$

O^{EC} is the domain ontology model connected with the conceptualisation of the notion "Electronic Commerce" and O^{Search} is the ontology model of information retrieval task then the model of inference machine can be redefine as it will be shown below. We can also to expand the set $\{O^{d\&t}\}$ by the user's model and receive personal-oriented ontological system model.

We can build the ontological systems typology and discus the formal properties of its members but it is the theme of a separate paper.

Let's consider how an ontological model works in specific domain.

3.2. Ontological System Model: an Example

In according to processing of Web-resources goal we need:

- domain and meta knowledge connected with the knowledge space under design;
- knowledge based inference engine and
- intelligent communication engine.

Domain and meta knowledge representation are the ontology of specific domain and tasks under design and meta ontology. Inference engine based on production-frame knowledge representation formalism is used at the stages of users requests processing and ontologies processing. And intelligent communication engine is responsible for the user's queries processing.

So let's discuss a model example connected with the gathering information from Web about electronic commerce. We need in appropriate domain ontology for solving the task (cup of it is depicted on Fig. 1).

The specification of part of domain ontology may be the following:

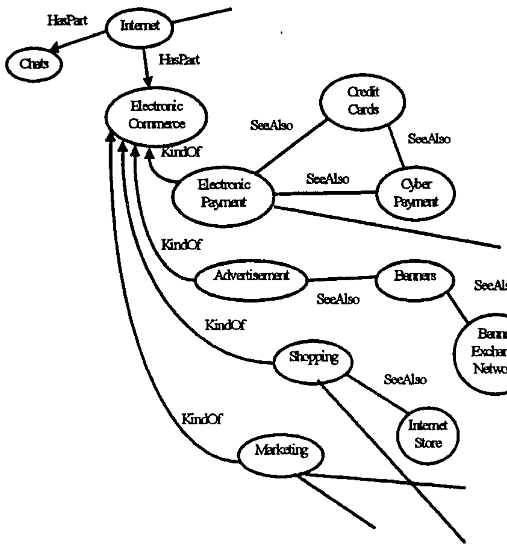


Fig. 2. Domain ontology "Electronic Commerce" fragment

```
[EC_Ontology is_a Ontology;
  Nodes = { Internet, Chats,
          ElectronicCommerce, CyberPayment,
          Advertisement, ... };
  Arcs = {HasPart, KindOf, SeeAlso ... };
  Inference = EC_DomainInferenceMachine ()];
[CyberPayment is_a Concept;
  State = "nonprocessed";
  Links = { KindOf#001, SeeAlso#001,
            SeeAlso#002 };
  URL = { "http://www.cyberplat.ru/", ...};
  Interpretation = IntConcept ()];
[SeeAlso is_a Relationship;
     FullName       = "SeeAlso";
     Interpretation = IntSeeAlso ()];
[SeeAlso#001 is_a SeeAlso;
     Source         = CreditCards;
     Destination    = ElectronicPayment;
     State          = "nonprocessed"];
.........................................................
```

We present the fragment of domain ontology where, in particular, the relationship SeeAlso is specified. It activates when the value of *State* slot changes and the demon of *if_changed* type with the name *IntSeeAlso* is started. The last is the following production system:

```
SeeAlso () {
  rule SLEEP
     :: [$CurrSeeAlso: State] ==
     "nonprocessed"
     ➔ exit ();
  rule NEW_TOPIC_SEARCH
     :: [$CurrSeeAlso: State] == "active"
     ➔ $Topics = [$CurrSeeAlso: Destination];
        [$Topics : State] = "active";
        write ($Topics); exit ();
  rule NEW_TOPIC_IS_NOT_RELEVANT
     :: [$CurrSeeAlso: State] == "passive"
     ➔ exit ();
}
```

In addition to domain ontology we need in task's ontology that can be specified as the single node *Search* for the simplicity:

```
[TaskOntology is_a Ontology;
   Nodes = { Search, ... };
   Arcs = { Demands, DoAlso, ... };
   Inference = TaskDomainInferenceMachine ()];
[Search is_a Concept;
   State = "nonprocessed";
   Links = { Demands#001, DoAlso#001, ...};
   Interpretation = IntSearchConcept ()];
```

We presented above the ontologies of domain level. But in our ontological system the meta ontology was depicted too. Our approach to meta ontology description is based on heterogeneous semantic networks (HSN) formalism presented in [19]. The HSN specific is (first of all) in explicit outline of semantically links and in their typology based on lexical values of natural language predicators and features of links (regularity, transitivity, etc.). Main types of semantically links outlined within HSN-theory for natural (Russian) language with their features are depicted in Tab. 3. All these semantically links can be divided into 9 types with the same features within each of them.

Table 3.

Kinds of semantic links	Links features values									
	Tr	Ntr	Atr	Rf	Nrf	Arf	Sm	Ans	As	Ns
Gen			+			+				+
Des		+			+				+	
Ins		+			+				+	
Cous	+				+				+	
Com	+			+				+		
Cor		+		+			+			
Lim	+					+			+	
Med		+			+				+	
Neg		+				+	+			
Pos		+			+				+	
Pot		+			+					+
Res	+				+				+	
Rep	+				+				+	
Sit	+					+			+	
Dir	+					+			+	
Trg	+				+				+	
Fin	+				+					+

The main benefit of HSNs using as top level ontology is the reducing of inference complexity due to meta level inference.

Let's specify, for example, some semantic links from top ontology useful for our task. Among others these are *Gen, Lim, Fin* from Tab. 3.

We should define the prototype of HSN semantic link notion at first. It is clear that it will be the following instantiation of the notion *Relationship* presented above (all HSN-links features in the definition are presented by slots with the type *Boolean* and named as in Tab. 3:

```
[HSN_link is_a Relationship;
  Tr, Ntr, Atr, Rf, Nrf, Arf,
   Sm, Ans, As, Ns          bool;
                            by_default false];
```

Then (to take into account previous definition) semantic link *Gen* that can be reflected by surface phrase "object A belongs to the set M" and is non-transitive (*Atr*), non-reflexive (*Arf*) and non-symmetrical (*Ans*) has the following definition:

```
[Gen is_a HSN_link;
        Atr = true;
        Arf = true;
        Ans = true];
```

In the same manner:

```
[Lim is_a HSN_link;    [Fin is_a HSN_link;
        Tr = true;              Tr = true;
        Arf = true;             Ntf = true;
        As = true];             Ns = true];
```

If the properties of the relationships in domain ontology are determined we can "understand" with what type of relationship it is necessary to work in top ontology.Interrelations between all above discussed ontologies depicted on Fig. 2.

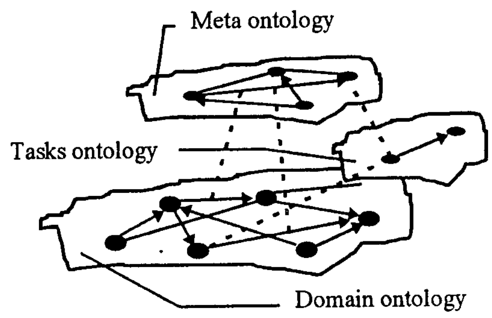


Fig. 2. Domain, tasks and meta ontologies interrelations

3.3. Top View at How an Ontological System Works for Web-resources Processing Domain

Let's suppose that the user's request is *"The ways of advertisement on Internet?"*.

Pre-processing of query gives us the following list of keywords: *way(s), advertisement, Internet.* Each of them should be mapped onto the domain ontology concepts (via Stop Vocabulary, for example). Several nodes should be "flash" according to ontology presented above. Otherwise our ontology is not complete and special procedure of its update should be run. There are two concepts in our ontology become active: *Advertisement* and *Internet*. That's why, inference machine associated within our ontology system should process them by the following manner:

- due to the processing the relationships HasPart (between concepts Internet and Electronic Commerce) and KindOf (between concepts Advertisement and Electronic Commerce) the last concept "flash" too;
- processing of the concept Electronic Commerce (via processing the relationships KindOf) leads to the activation of the concepts Electronic Payment, Shopping, Marketing, etc. and, at least,
- activation of the relationships SeeAlso gives the activation of the following concepts: Credit Cards, Cyber Payment, Banners, Banners Exchange Network and Internet Store.

On the finishing of the "wave" inference within the domain ontology we'll receive the updated list of key concepts (not keywords!): Internet, Electronic Commerce, Advertisement, Electronic Payment, Shopping, Marketing, Credit Cards, Cyber Payment, Banners, Banners Exchange Network and Internet Store. The specifications of those concepts are scanned for the extracting the values of slot URL (if such slots are present in specifications and have values).

Let's suppose that only the following concepts posses the values of slot URL:

- Marketing (http://www.citforum.ru/marketing/index.shtml, http://www.marketing.spb.ru/, etc.),
- Cyber Payment (http://www.cyberplat.ru/, etc.) and
- Internet Store (http://www.cd.ru/,

http://bshop.ipassage.ru/, etc.).

All of them excluded from the further key terms list processing and the values of their URL give us the header part of the response for user request.

There are two strategies for the processing of other key terms. The first one is connected with the updating current domain ontology via dialogue with user and its discussion is out of paper. The second is connected with the ranking the other key terms according to their generality (specific concepts have the highest priority). These terms Internet and Electronic Commerce are so general that can't bring the concrete information. At the finishing stage of users request processing appropriate query is generated for the search engines working on Web.

The results of the search are processed too. There are two approaches that can be used for the analysis of the results and decision about relevance of the received documents. The first one lies in the comparison of URLs presented in the documents with the URLs already presented in current ontology. The last is little bit sophisticated and supposes using of bootstrapping like procedure. We consider that annotation of each received document is the new quasi-request. So we can apply to it the described processing procedure and receive its new quasi-pattern. Then we can compare this one with the initial pattern of user's request. If they are the "same" the result document is relevant to the initial request. Unfortunately, we have no room for the deep discussion of interesting and very important topic in the paper.

4. FireExpert: Multiagent System for Intelligent Processing of Web-resources

Multiagent approach had been chosen for the design, development and implementation of FireExpert system. The main goal of the system is search and gathering of information resources on Web. So FireExpert should solve the following generic tasks:

- ontology-based specifying of the users requests;
- standard search engines using;
- processing of search results;
- analysis of the received documents for their relevance to initial requests;
- informational resources thematic bases support
- users' ontologies support.

4.1. FireExpert System Architecture

The hybrid and distributed architecture is chosen for FireExpert design and implementation. It is possible to pick out three logical levels in system's architecture: interface, monitoring and meta level. The ontologies models implementation and Ontology Service Agents belong to the meta level. Informational, Intelligent and Auxiliary Agents are at the monitoring level and Interface Agents are at the interface level.

To satisfy the modern distributed Internet applications requirements FireExpert (FE 1.0) system architecture is based on the distributed client-server architecture with "thin" clients. Its common structure depicted on Fig. 3.

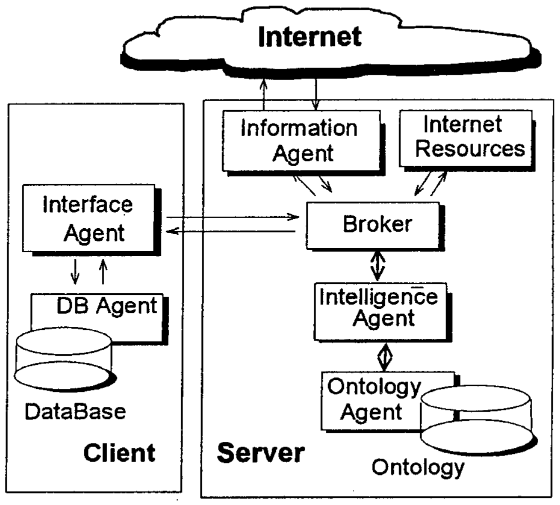


Fig. 3. FireExpert architecture.

The basic functions of client and server parts are presented in Tab. 4, 5.

Table 4

Server part agent functions	*Operations*
Inquiry processing control	Receiving inquiry from client
gathering information from Web	Access to searching engines
Analysis of acquired information	Ontology based processing of the results
Ontology storing	Ontology forming
DB inquiries storing	DB inquiry forming

Table 5

Client part agent functions	*Operations*
Inquiry forming	Transmission of the inquiry to the server
Forming (editing) the ontology	Transmission of an ontology scheme to server

Forming (escorting) document base to user	Processing of the result database

As follows from the above tables, all main tasks are concentrated at server in FireExpert system. It gives us fast working applications, an easy control over the system on the whole and small-size client part ("thin" client).

Server is capable of working in three modes:

- Mode of executing of the users inquiry. User (client) may send query to server from the interface of a program (separate Java application or HTML - interface, which is loaded to web-browser). The TCP/IP protocol is used for data transmission.
- Mode of ontology supporting. The users (clients) are working with ontologies (data receiving, editing, etc.) at client. All ontology changes are transmitting to server by the TCP/IP protocol and are saving.
- Mode of ontology supplementation. User may specify mode in which agents use the existing ontology components and regularly search for the information (every day, week, or month) in web, thus expanding the ontology.

Formal specification of the system architecture may be performed with the use of the same production-frame formalism [27], which was discussed above in the section devoted to the ontology specification. To demonstrate such a specification technique, we will consider a fragment the intentional representation of the architecture of the FireExpert 1.0 system.

First of all, we need to specify the following set of prototypes that fix the structure of an arbitrary system:

```
[System is_a prototype;
    consist_of {frame}, restr_by Component; ...];
[Component is_a prototype;
    consist_of {frame}; ...];
```

In addition, we'll define the structure of a multiagent system with a client-server architecture by using the following set of prototypes and examples:

```
[ ClientServerMAS is_a System;
        consist_of = {ServerPart, ClientPart} ... ];
[ServerPart is_a Component;
        consist_of {frame}, restr_by Agent; ... ];
[ClientPart is_a Component;
        consist_of {frame}, restr_by Agent; ... ];
[Agent is_a prototype; ...];
```

The descriptions presented are not yet related to the FireExpert system in any way. In order to fix such a relation explicitly, we specify the presented conceptions as follows:

[FireExpert is_a ClientServerMAS; ...];

By using this specification we state that the FireExpert system can be classified as a Client - Server MAS system. Consequently, it contains client and server parts, which in turn consist of agent components.

In the following subsection we will show how the further specification of the FireExpert system can be realised. Now we only conclude that formalism used is rather flexible and powerful to specify a wide class of systems.

4.2. FireExpert Agents Specifications

According to the generic tasks listed above, we will consider the following agent types:

- Interface agents,
- Informational agents,
- Intellectual agents,
- Ontology agents,
- Internet-resources agents,
- Database agents.

Interface agents are responsible for communication with users, receive the user's inquiries and transmit them to informational agents. They are also used in co-operation with searching engines and for receiving the results of the processing.

Informational agents are intended to retrieve the documents in the Internet. Transmission of the user's inquiries to the searching engines (using the formats compatible to particular searching engines) and receive the searching results.

Intellectual agents process the results from informational agents in order to determine the relevancy of a document to the user's inquire. For this purpose they co-operate with ontology agents.

Ontology agents are support output and provide access to ontologies also for it's renewal.

Database agents co-operate with the relational databases.

Internet-resources agents copy the HTML documents from the web then scan them to exclude obsolete and repeating references and analyse the structure of the HTML documents.

Knowledge that possesses the agents of a system may be structured by the following way: searching for the common distributed knowledge of the agents, particular knowledge and searching for necessary resources. Knowledge necessary for each type of agent is shown in Tab. 6. However the agent that co-ordinates transmission of the input/output data was brought in, in addition to agent types listed above.

Table 6

Agent type	Partial knowledge	Shared knowledge	System resources
Interface agents	User's settings	Ontology scheme	HTML document format
Informational agents	Knowledge of inquiry formats for different types of search engines, its Internet addresses, protocols, etc.		
Intellectual agents	Algorithms for comparison and making a decision wherever the document relevant or not	Ontology	HTML document format
Ontology service agents	Methods for data extraction t from the ontologies and for editing and replacement of the ontology	Ontology, ontology scheme	
Database agents			Results database
Internet-resources agents	Data transmission protocols		Information about HTML documents
Broker agent	Data transmission protocols		

There is no central control for the agents in FireExpert system. Network communications are supported by Java Remote Method Invocation. General scheme of agent's interactions depicted on Fig. 4.

It is clear that the agents specification is based on the definition of the following notion:

```
[Agent is_a prototype;
   ..............................
   PartialKnowledge {frame};
   SharedKnowledge {frame};
   SystemResources  {frame};
   CommunicateWith {frame}, restr_by Agent];
```

According to the FireExpert system agents types it is possible to introduce the specifications of

```
[InterfaceAgent is_a Agent;
   PartialKnowledge = {UserModel};
   SharedKnowledge = {TopOntology,
                      TasksOntology,
                      DomainOntology};
   SystemResources  = {HTML-format};
   CommunicateWith = {dbAgent, Brocker}];

[InformationalAgent is_a Agent;
   PartialKnowledge =YahooQueryFormat,...};
   CommunicateWith = {Brocker, ...}];

[IntelligentAgent is_a Agent;
      PartialKnowledge = {InferenceEngine,...};
      SharedKnowledge = {TopOntology,
                         TasksOntology,
                         DomainOntology};
      SystemResources  = {HTML-format};
CommunicateWith = {OntologyServiceAgent,
                   Brocker}];

[OntologyServiceAgent is_a Agent;
      PartialKnowledge = {EditEngine,
                          AsquisitionEngine,...};
      SharedKnowledge = {TopOntology,
                         TasksOntology,
                         DomainOntology};
      CommunicateWith = {IntelligentAgent}];

[InternetResourcesAgent is_a Agent;
      PartialKnowledge = {TCP/IP, HTTP,...};
      SystemResources  = {HTML-format};
```

CommunicateWith = {Broker,...}];

....

And to determine (to take into account the data from Fig. 4 and Tab. 4, 5) FireExpert system server and client parts as follow:

[ClientPartFE is_a ClientPart;
consist_of = {QueryAgent, EditAgent,...};...];
[ServerPartFE is_a ServerPart;
consist_of = {Co-ordinationAgent,
ProcessQueryAgent,
InternetResourcesAgent,
OntologyServiceAgent, ...}; ...];

In such a manner it is possible to expand the agents specifications to the level needed for their implementation.

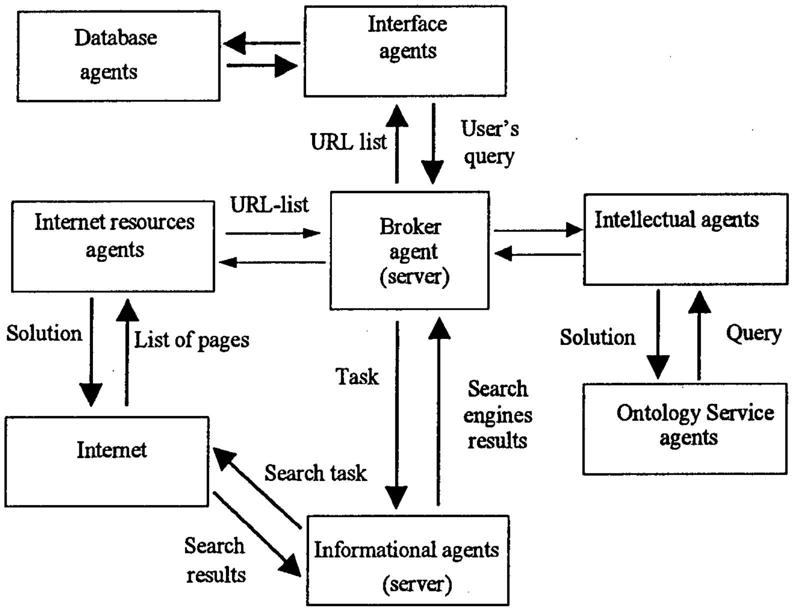
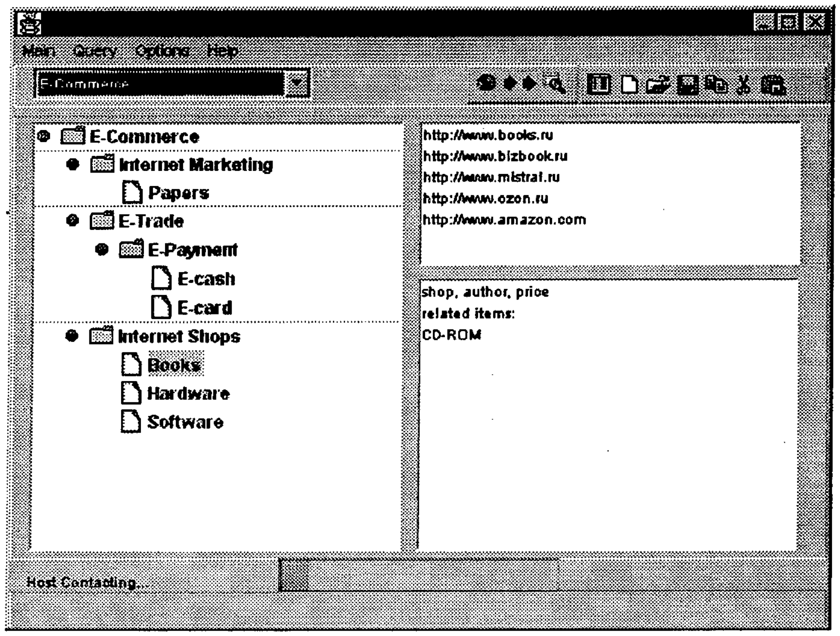


Fig. 4. Agent's interactions diagram

To illustrate the technique of such an expansion let's discuss the description of FireExpert system interface agent screen form (Fig. 5).



http://www.books.ru
http://www.bizbook.ru
http://www.mistral.ru
http://www.ozon.ru
http://www.amazon.com

shop, author, price
related items:
CD-ROM

Host Contacting...

Fig 5. FireExpert system interface agent screen form

There are three main areas in the screen form: pull-down menu and button-bar where presented buttons duplicate options from menu and drop-down list with ontology under processing, work table and status line. Worktable is divided into three windows. The first one used for the representation of current ontology scheme as quasi-tree (course in fact the ontology is not tree but network). Two others reflect the current notion attributes - the URL-list connected with the notion and selected notion and its relationships description.

To support the screen form implementation first at all it is necessary to expand above presented *InterfaceAgent* specification with the slot *ScreenForm* of frame type. And further, to add into specification knowledge base the set of the following prototypes:

```
[ScreenForm is_a prototype;
    MainWindow  frame, restr_by Window;
    ConsistOf      {frame} ];

[Window is_a prototype;
   Caption       string;
   X0, Y0        int;
   W, H          int;
   ConsistOf   {frame}... ];

[ Menu is_a prototype;
   Options       {frame}  ];

[ PopupOption is_a prototype;
    Title string;
    Options        {frame} ];

[ ItemOption is_a prototype;
    Title string;
    ShortKey      string;
    Support        frame ];

[ToolBar is_a prototype;
    Options        {frame}, restr_by {Button, ...} ];

[Button is_a prototype;
    Icon           string;
    Support        frame ];

[WorkTable is_a Window];

[StatusLine is_a Window];
....................
```

FireExpert screen form specification based on above depicted prototypes can be described as follow:

```
[ScreenFormFE is_a ScreenForm;
    MainWindow  = MainWndFE;
    ConsistOf        = { MenuFE, ToolBarFE,
                          WrkTblFE, StatusLineFE } ];

[MainWndFE  is_a Window;
    Caption = "Intelligent System FireExpert 1.0";
    X0 = 0; Y0 = 0; W = 400; H = 300 ;
    ConsistOf = {WndOntologyDescr, WndURL,
                        WndNotionDescr}];
```

```
[WndOntologyDescr is_a Window;
    \CurrOntology frame, inher_from
                              OntologyDscr];

[MenuFE is_a Menu;
   Options = {Main, Query, Options, Help}  ];

[ Main is_a PopupOption;
   Title= "Main";
   Options     = {New, Load, Save, ...} ];

[ New is_a ItemOption;
   Title= "New";
   ShortKey    = "Cntr + N";
   Support     = NewOntologyMethod ];
....................................................
```

Expanding of specifications till the bottom level of all frames examples we can receive full-scale specification of FireExpert screen form. Some values of slots in specification will represent the parameters (for example, string "Intelligent System FireExpert 1.0" is the *Caption* slot value and on the other hand it is factual parameter of Window class constructor). Others will represent the implementation of the methods connected with appropriate objects (for example, frame NewOntologyMethod is the *Support* slot value and on the other hand it is Java code support the new ontology creation).

5. Conclusion Remarks and Future Developments

An approach to intelligent Web-resources processing in spirit of ontology context and multi - agent support was discussed. Full scale development and implementation of presented approach is connected with the solving of the following problems:

- pre-processing of the users queries at the base of NLP techniques for the receiving as an output their full scale semantically representation;
- researches in the domain of relevance problems;
- complete implementation of presented multi - agent system FireExpert.

All of them in the hot spots of our research connected with the development of the user-friendly tools for the design, maintenance and processing of WEB-oriented ontology based multi - agent systems.

Acknowledgements

Presented research partially supported by Russian Foundation for Basic Researches (Grant N 99-01-00844).

Bibliography

1. Benjamins V. R., Fensel D., et. all, Community is Knowledge! In KA2, KAW'98, Banff, Canada, 1998.
2. Brazier, F.M.T., Dunin-Keplicz, B., Jennings, N.R., and Treur, J., Formal Specification of Multi-Agent Systems: a Real World Case In: Lesser V. (ed.), Proceedings of the First International Conference on Multi-Agent Systems, ICMAS'95, MIT Press, Menlo Park, VS, 1995
3. Carol E.Brown, Les Gasser, Daniel E. O'Leary, and Alan Sangster, AI on the WWW: Supply and Demand Agents, IEEE Expert 10(4): 50-54 (1995)
4. Chandrasekaran B., Josephson J.R.,Benjamins R., The Ontologies of Tasks and Methods, KAW'98, Banff, Canada, 1998.
5. Craven M., DiPasquo D., et al., Learning to Extract Symbolic Knowledge from the World Wide Web, Submitted to AAAI-98. January 1998. http://www.cs.cmu.edu/afs/project/theo11/www/wwkb/
6. Farquhar A., Fikes R., & Rice J., The Ontolingua Server: A Tool for Collaborative Ontology Construction. Knowledge Systems Laboratory, KSL-96-26, September 1996.
7. Fensel V. D., Decker S., Erdmann M., Studer R., Ontobroker: Transforming the WWW into a Knowledge Base, KAW'98, Canada, April, 1998.
8. Fridman N., Hafner, Ontology Design: A Survey and Comparative Review, AI Magazine, 18 (3):Fall 1997
9. Guelfoyle W., Intelligent Agents: the New Revolution in Software, Ovum. IBM Agents, http://activist.gpl.ibm.com:81/WhitePaper/ptc2.htm (1994).
10. Human Agent Interaction, FIPA98 Draft Specification: Part 8, http://www.cset.it/fipa/
11. Iglesias C. A., Garijo M., Gonzalez J. C., Velasco J. R., 1996, A Methodological Proposal for Multiagent Systems Development extending CommonKADS, In Proccedings of KAW'96 Banff, Canada, 1996.
12. Jonker, C.M., and Treur, J., A generic multi- agent architecture for interactive diagnostic tasks with clarification. In: J. C uena (ed.), Proceedings of the IFIP'98 Conference on Information Technology and Knowledge Systems, IT&KNOWS'98, Chapman and Hall. 1998.
13. Khoroshevsky V. F., Knowledge v.s. Data Spaces: How an Applied Semiotics to Work on Web, In Proc. CAI'98, Puschino, Russia, (1998).
14. Kifer, M., Lausen, G., and Wu, J. Logical foundations of object-oriented and frame-based languages. Journal of the ACM, 1995.
15. Lenat, D. B., CYC: A Large-Scale Investment in Knowledge Infrastructure, Communications of the ACM 38, no. 11 (November 1995).
16. Lewis L., AI and intelligent networks in the 1990s and into the 21st century, In: Liebowitz, J. & Prereau, D. "Worldwide intelligent systems", IOS Press, (1995).
17. Luke S., Spector L., Rager D., Hendler J. Ontolodgy-based Knowledge Discovery on the World-Wide-Web, In the Proceedings of the Workshop on Internet-based Information Systems, AAAI-96, Portland, Oregon, 1996
18. Ontology Service, FIPA98 Draft Specification: Part 12, http://www.cset.it/fipa/
19. Osipov G.S., The Method of Direct Knowledge Acquisition from Human Experts, In Proccedings of the 5th Banff

Knowledge Acquisition for Knowledge-Based Systems Workshop Banff, Canada,November 1990.
20. R. Kumar, Internet Information Resource Discovery Tools: Current Status and Future Trends, In the Proc. of the CIKM'95 Intelligent Information Agents Workshop, Baltimore MD, December, 1995.
21. TOVE Manual: Department of Industrial Engineering, University of Toronto, http://www.ie.utoronto.ca/EIL/tove/Onto!OC.html
22. Woods W. A. et. al., Conceptual Indexing for Precision Content Retrieval, http://www.sunlabs.com/research/knowledge/
23. Agent Builder White paper, http://www.agentBuilder.com/
24. Etzioni O., Weld D., 1995 IEEE Expert, July 1995.
25. Franklin S., Graesser A. "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996
26. Nwana H. S. "Software Agents: An Overview", Knowledge Engineering Review, Vol. 11, No 3, pp.1-40, Sept 1996. Cambridge University Press, 1996
27. Khoroshevsky V.F. "Knowledge Based Design of Knowledge Based Systems in PiES WorkBench", In Proc. Of Japan-CIS Symposium on Knowledge Based Software Engineering'94, Pereslavl-Zalessky, Russia, 1994
28. Khoroshevsky V.F., "Knowledge Based Design of Intelligent Interfaces in PiES WorkBench", In: Proc. of the Sixth International Conference on Artificial Intelligence and Information-Control Systems of Robots, Smolenice, Slovakia, September 21-25, 1994, World Scientific, 1994
29. Braetman J.A., Magnini B., Rinaldi F. The Generalized Italian, German, English Upper Model, ECAI'94, Amsterdam, 1994
30. Karp P.D., A Qualitative Biochemistry and Its Application to the Regulation of the Trytophan Pperon. In Artificial Intelligence and Molecular Biology, ed. L. Hunter, AAAI Press, 1993
31. Van der Vet P.E., Mars N.J., Speel P.H. The Plinius Ontology of· Ceramic Materials, ECAI'94, Amsterdam,1994

APPLICATION OF HYBRID AGENT-BASED TECHNOLOGY FOR DESIGN OF TELECOMMUNICATION SYSTEMS

Igor V. Kotenko

Department of Telecommunication Systems, St.-Petersburg Signal University.
E-mail: ivkote@robotek.ru

Abstract

The paper describes the basic results of creation of the multi-agent intelligent computer-aided System for Design and development Planning of Telecommunication Systems (SDPTS): (1) principal theses of the conception of SDPTS construction as a hybrid multi-agent system; (2) general models of design processes as integrated models of SDPTS software agents' community functioning; (3) models of design objects used for SDPTS agents' reasoning about subject domain; (4) models and algorithms of SDPTS agents' functioning; (5) software prototypes of basic agents and their efficiency evaluation.

Keywords: *computer-aided design, multi-agent systems, intelligent software agents, computer-supported cooperative work, telecommunication systems.*

1 Introduction

Now we can establish that the volume, complexity, distribution and change dynamics of information necessary for design and development planning of telecommunication systems essentially have increased. Also the requirements of the efficiency of these processes and quality of accepted decisions considerably have grown. However the used technology of design and development planning of telecommunication systems is characterized by high laboriousness, large temporary expenses and unsatisfactory decision validity.

The existing works on automation of telecommunication system design and development planning are in the undeveloped condition. As a rule these works are focused on a local task-oriented approach and don't use uniform technology.

The majority of the systems for automation of given processes, developed hither to, has a number of essential lacks, that adds one more level of complexity and results in natural refusal of their use: (1) designers should work in the strictly certain place; (2) design process is automated only partially; (3) stages of conceptual design are not supported; (4) subject domain models are not interconnected; (5) management facilities of design technology are not provided; (6) systems are "obtrusive", ordering the rigidly given design order; (7) methods which have not confirmed the adequacy are used; (8) the realized ways of interaction of external environment and designers with systems do not correspond to the requirements; (9) the very weak opportunities of task solvers and automatic schedulers are realized only; (10) designers' actions are not supervised, since there are no means of prevention, detection, criticism and correction of mistakes; (11) mechanisms of cooperative work support are absent or are not advanced and etc. ([Kotenko et al., 1996] [Kotenko, 1998a]).

One of the most cardinal ways of an output from an existing situation is an automation of the given processes based on "new" information technologies. This means the necessity of creation of the distributed continuously evolved intelligent computer-aided systems for design and development planning of telecommunication system (SDPTS), which elaboration is conducted on the basis of uniform construction conception founded on multi-agent technology ([Wooldridge et al., 1995], [Gorodetski, 1996], [Wooldridge et al., 1998], [Brenner et al., 1998]), integrated set of models of the telecommunication systems and processes of their functioning, design process models, methods, models and algorithms realizing a complex automated transformation of design information.

SDPTS must support the developers in performing their tasks and increase their understanding and competence to a level that allows them to make justifiable decisions, using all intelligent design agents that are friendly to users and available on all computer sites in large computer networks used for design and development planning. The SDPTS realization will allow to use the new technology of computer-aided design and planning of telecommunications uniting a complex of methods, ways and means of automated multi-agent processing, transfer, storage and display of information on all the stages of design and planning.

2 Conception of SDPTS construction

2.1 General theses on SDPTS construction and design process realization

According to offered conception *the essence of design process*, which is carried out by the use of SDPTS, consists in realization of automatic and automated operations of transition from general representations of telecommunication system to detailed development of its components, consecutive detailed elaboration of telecommunication system elements, and then combining them in a single unit - telecommunication system as the whole and its optimization [Kotenko, 1998a].

The scheme of realization of automated design processes is set as follows: the designers, working in computer networks (LAN - MAN - WAN, or Intranet - Internet), manipulate various objects, exchange different data, start automatic procedures (for example, for testing variants of resource distribution etc.). SDPTS continuously watches designers' actions and "unobtrusive" directs them acting in a role of adviser.

SDPTS is built as a distributed multi-agent system, described by the following model (fig.1):

$$M_{SDPTS} = < \{A_j\}, \{T_k\}, \{TA_i\}, \{TAR_m\}, \{CS_i\}, \{CP_n\}, \{CR_r\}, \{E_s\}, \{S_t\}, \{FA_h\} >,$$

where $\{A_j\} = < N_j, D_j, R_j, M_j, V_j >$ - agents' types (N_j and D_j - name and area of agent's competence, R_j - resources, required for agent's work, M_j - agent's metaknowledge (metaalgorithm of functioning), V_j - additional attributes describing the agent); $\{T_k\} = < N_k, P_k, M_k, I_k, O_k, P_k >$ - types of automated design tasks (N_k - task name, P_k - task purpose, M_k - decision model describing decomposition of a task on subtasks and sequence of their decision process, I_k and O_k - input and output data, P_k - parameters of decision efficiency); $\{TA_i\}$ - function of distribution of tasks between agents; $\{TAR_m\}$ - resource restriction (including temporary and computational) on process of realization of tasks by agents; $\{CS_i\}$ - strategies of cooperative work and agents' cooperation during design; $\{CP_n\}$ - protocols of agents' negotiation; $\{CR_r\}$ - protocols of avoidance, detection and resolution of conflicts between various agents' actions; $\{E_s\}$ - events caused by actions of agents and external influences; $\{S_t\}$ - SDPTS states, determined by states of agents $\{A_j\}$ and tasks $\{T_k\}$; $\{FA_h\}$ - transformation of states arising as a result of actions of agents and external events, and resulting in evolution of design objects (elements of telecommunication system) [Kotenko et al., 1994c].

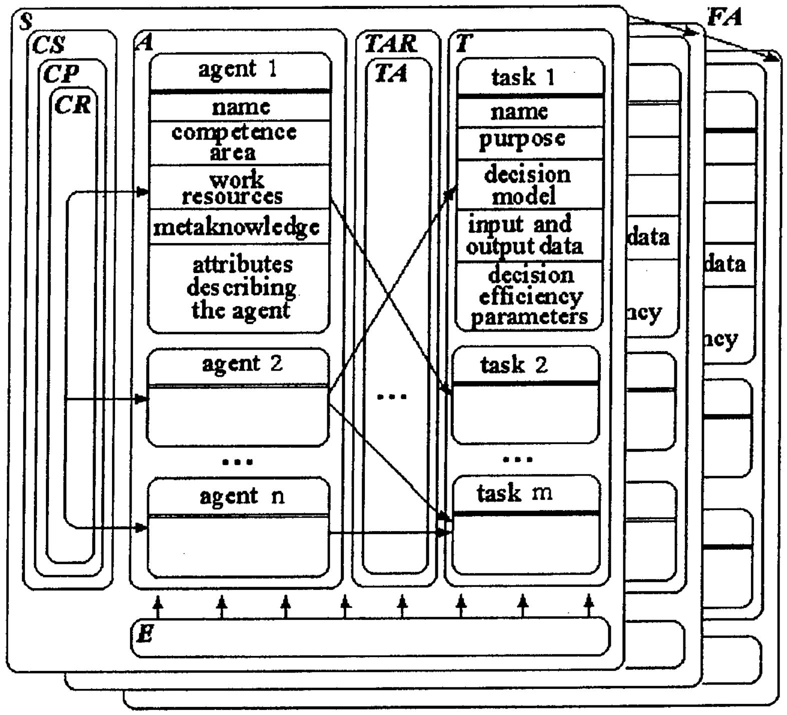


Fig.1. Common architecture of SDPTS as multi-agent system

The scheme of solving a concrete design task (process) by the given approach use is the following: (1) subject domain knowledge, knowledge of restrictions and requirements to design objects is transferred to agents interested in them; (2) each agent solves a fixed task; (3) locally effective decisions are transferred to specially allocated agents; (4) after completion of decision formation cycle a result is prepared for dispatch on interested agents for analysis; (5) all the process is repeated before achievement of global effective solution. The generalized SDPTS structure is based on principles of sophisticated local control and organizational structuring [Durfee et al., 1989]. The most important agents' classes are the ones of *agents' coordination, cooperation and collaboration support*. They owe: (1) to represent competitive actions; (2) to describe temporary relations between actions; (3) to process uncertain and incompletely developed agent's plans; (3) to suppose several levels of action plan abstraction; (4) to take into account both negative and positive relations between the local plans of various agents; (5) to provide agents' interaction, and etc.

The modeling approach of SDPTS construction assumes its development as a *system of hybrid agents* uniting performance of traditional "rigid" algorithms, manipulation of facts in a database, modeling by means of various imitating and analytical models from model base, processing of knowledge from knowledge base and use of telecommunication system projects already realized and solved tasks from precedent base [Kotenko et.al., 1994c].

Basic principles of automated design realization are: (1) granting to the designers of functionally complete set of opportunities that ensure the realization of various design operations; (2) integrativity; (3) intellectuality; (4) interactivity.

2.2 Base SDPTS features

The *base SDPTS features* are: (1) realization of design as operations of detailed elaboration of telecommunication system elements; (2) unity of all the stages of information process; (3) "uniform" information base; (4) design decision support; (5) openness and adaptability; (6) distributed cooperative group work support; (7) support of case-based design; (8) support of various modes of operations and interactions of users with SDPTS; (9) automated document workflow; (10) support of hypermedia information manipulation; (11) processing of incomplete and contradictory information; (12) version and development history support; (13) information reliability support; (14) maintenance of design guaranteed time; (15) designers' tutoring and work accompaniment support; (16) information security support and etc.

The basic principles, methods and strategies of intelligent cooperative design are: (1) hierarchical parallel multilevel design, (2) generation and testing of suggestions, (3) combination of least-commitment and intelligent casual-commitment, (4) combination of linear and nonlinear design action planning, (5) opportunistic design action planning, (6) constraint propagation, (7) temporal and resource planning, (8) meta-level reasoning, (9) skeleton (precedents) design (case-based reasoning), (10) qualitative reasoning [Kotenko, 1995b].

2.3 Methods of automated design and planning realization

The general family of methods of automated design and planning realization is characterized as set of methods of the *cycle "task definition → variants generation → results explanation → analysis, estimation, selection and criticism of decision variants → variants modification → SDPTS agents' learning"* (fig.2).

The basic methods of decision variants generation are the methods of (1) decomposition - assembly, (2) transformation, (3) case-based reasoning, (4) constraint propagation and (5) classical methods of operations research.

Proceeding from definition of methods, used for solving of each type of design tasks, both knowledge and subtasks, necessary for realization of each method, based on [Chandrasekaran, 1986], *design task structure* is constructed.

One of the main components of SDPTS construction conception is the offered *approach of solution generation by means of derivational analogy*. This approach allows to carry out design processes by repeated fulfillment of modified decision plans of design tasks realized, basing on their representation as hierarchical structure of goals and on the new tasks solution by descending exposure of these plans [Kotenko, 1998b].

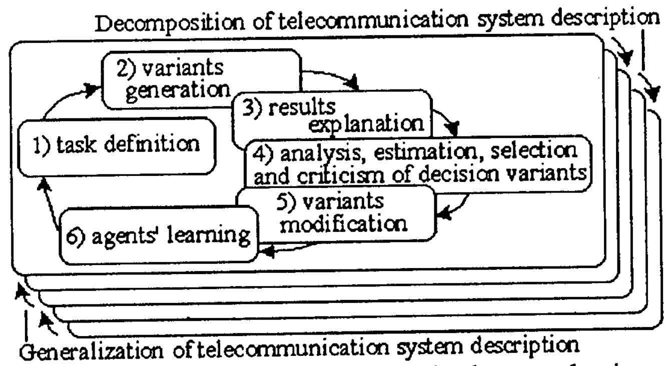

Decomposition of telecommunication system description
2) variants generation
3) results explanation
1) task definition
4) analysis, estimation, selection and criticism of decision variants
5) variants modification
6) agents' learning
Generalization of telecommunication system description

Fig.2. Main stages of automated design development planning

2.4 SDPTS architecture and general design algorithms

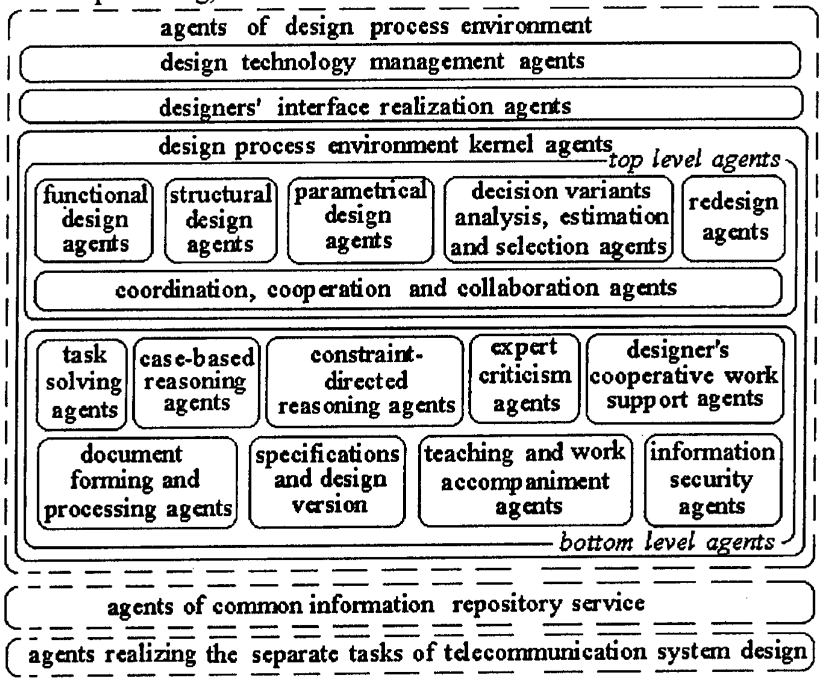
The basic theses of SDPTS construction conception are: (1) definitions of basic concepts of automated design, (2) scheme of use of traditional and expert SDPTS agents (deliberative, reactive and hybrid agents), (3) generalized algorithm of automated design, (4) structure of SDPTS as a multi-agent system, (5) algorithms of designers' work, (6) algorithms of the automated functional, structural and parametrical design, analysis, estimation and selection of decision variants, redesign and coordination of decision variants. The suggested strategy of the automated design, following from the given scheme of various agents' use, gives an opportunity of parallel performance of design processes by means of (a) traditional algorithmic methods of telecommunication system synthesis and analysis and design management (based on the rigidly given algorithms); (b) methods of heuristic search and logic reasoning; (c) formalizations of design purposes and tasks, generation of variants, their estimation and selection.

SDPTS architecture has three *logic levels* (fig.3): (1) level of agents of design process environment, (2) level of agents of common information repository service and (3) level of agents realizing the separate tasks of telecommunication system design. *The basic agents of design process environment* are (a) design technology management agents (guaranteed time maintenance and design realization agents), (b) designers' interface realization agents and (c) design process environment kernel agents. *The kernel of SDPTS design process environment* is divided on the top and bottom levels. Structure of *the top level agents* includes the agents of functional, structural and parametrical design, agents of the analysis, estimation and selection of decision variants, redesign agents and agents of coordination, cooperation and collaboration. The top level agents use results of work of the bottom level agents. *The bottom level agents* are the agents of task solving, case-based reasoning, constraint-directed reasoning, expert criticism, designer's cooperative work support, document forming and processing, specifications and design version forming, teaching and work accompaniment, information security, etc. *The generalized automated design algorithm* is given as an iterative sequence of operations of functional, structural and parametrical design, analysis, estimation and selection of allowable decision variants, redesign, coordination of decision variants and selection of optimum (rational) variants.

The developed *algorithms of designers' work* are based on iterative performance of operations of data input and decision inquiries, formation of decision variants, editing of results, search of the decisions in precedent base, decision demonstration in a common context and reception of the remarks (offers), modeling of decision variants, estimation of the moment of the variant generation termination, utility function construction, ordering of the accepted alternatives, coordination of decision variants, avoidance, detection and resolution of conflicts, distribution of results.

2.5 Approach to realization of computer-aided design technology

The offered approach to the design technology realization is based on unification of base design operations (modules), which yet do not contain the information, dependent on technology, and also definition and adjustment of parameters of basic technology elements - (1) tasks, (2) task flows and (3) automated design process environment [Kleinfeldt et al., 1994]. Unification of various agents realizing the technology is based on concept of modular abstraction and is carried out by integration of modules and (or) their embedding. The most important aspects of technology management are: task abstraction realization, automatic performance and generation of flows, management of data changes and task flows, maintenance of convenience and simplicity of SDPTS use [Kotenko, 1998a]. Use of this approach allows to present SDPTS as mobile and well organized agent-based system, capable to adjustment on feature of a subject domain and on the varied requirements.


agents of design process environment
design technology management agents
designers' interface realization agents
design process environment kernel agents
top level agents
functional design agents
structural design agents
parametrical design agents
decision variants analysis, estimation and selection agents
redesign agents
coordination, cooperation and collaboration agents
task solving agents
case-based reasoning agents
constraint-directed reasoning agents
expert criticism agents
designer's cooperative work support agents
document forming and processing agents
specifications and design version
teaching and work accompaniment agents
information security agents
bottom level agents
agents of common information repository service
agents realizing the separate tasks of telecommunication system design
Fig.3. Basic SDPTS agents

2.6 Methodology of SDPTS development

According to the chosen evolutionary SDPTS development methodology for creation focused on the designers' need adaptive automated system, simply included on environment of various design processes realization, concrete SDPTS application should be formed by means of the special development support complex and evolutionary to be increased while in service. The development support complex is a set of SDPTS agents' generation tools, which by the use of the definition of the required design functions and tasks provide a choice of adequate design agents, their combination, adjustment and evolutionary adaptation to concrete application by means of appropriate formal and expert components.

3 Basis of the theory of computer-aided design and development planning of telecommunication systems

This theory (by analogy with [Tomiyama, 1986], [Ohsuga, 1989], etc.) is necessary for an all-round and "deep" substantiation of basic theses of SDPTS construction and computer-aided design realization, determining the design processes, objects and used means of knowledge representation and manipulation. The offered theory is given in a form of *uniform constructive as developed formal system*, concerning to a class of *"weak" theories*: it (a) *determines* by means of integrity restrictions a set of allowable decisions; (b) *accumulates* in a form of rules and precedents a designer experience; (c) for decision support *uses* pseudo-physical logics, soft models and algorithms, qualitative reasoning mechanisms and etc. *Given theory is based on three types of models*: (1) *models of processes* of design and development planning of telecommunication system, as integrated models of functioning of the SDPTS agents' community. They represent the design as "expansion" (in time and space) of telecommunication system elements expressing in consecutive definition of their structure and parameters by means of common resource distribution and fixing the results in a form of documents; (2) *models of telecommunication system elements* and factors, influencing them, necessary for realization of agent' reasoning about subject domain. These models represent structure and basic "laws" of functioning of telecommunication system, condition of its construction, opportunities and state of working telecommunication system as a design and planning resource; (3) *models of functioning of SDPTS intelligent software agents* (as design forces and means).

4 Formalization of computer-aided design and planning processes

The processes of the computer-aided design and development planning of telecommunication systems are submitted on the basis of the following *models*: (1) *prescriptive models* (showing, how design processes should be carried out, and determining a design methodology) and (2) *computing models* (specifying methods of data and knowledge representation and manipulation). In difference from works on the general design theory (for example, [Tomiyama, 1986], [Ohsuga, 1989], [Coyne, 1990]) the offered models in details describe concrete processes of design and development planning of telecommunication systems.

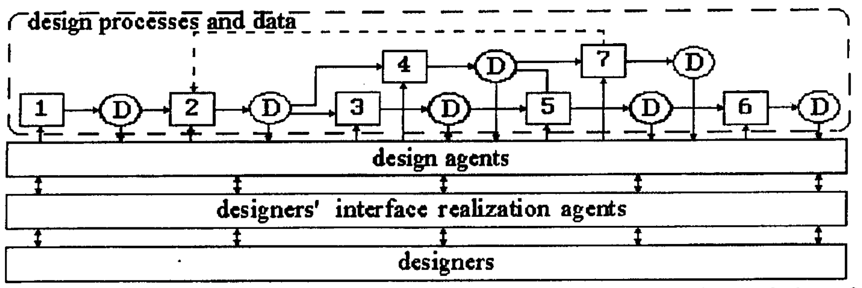
4.1 Prescriptive models of computer-aided design and planning processes

The prescriptive models include (1) general models of processes of design and development planning of telecommunication systems, (2) models of decision support processes, (3) models of documents and processes of their formation and processing.

The general models of design and development planning are given as sets of telecommunication system elements and their states, stages, processes, tasks, conditions and resources of the design and telecommunication organization, documents and relations between them, specifying the design restrictions and necessary design data transformations. For definition of separate design operations the description of their input, output, converter (technique, formalized or machine algorithm of operation performance), required resources and used intelligent software agents is given. The example of generalized graphical model of one of the design processes (efficiency evaluation and decision variant choice) is depicted in fig.4.

The models of decision support processes are given as (a) models of separate decision support processes and (b) integrated decision support model (as model of interconnected set of separate decision support processes).

Within the framework of *separate decision support processes* the basic decision support tasks of telecommunication system design and development planning are selected: (1) search of a route for message transfer ways; (2) allocation of a group not renewed resource in sole usage; (3) establishment of a sequence of shared resource use; (4) distribution of renewed resources; (5) evaluation of parameter values. The tasks are submitted by means of their name and statement attributes, elements


design processes and data
1
2
3
4
5
6
7
D
design agents
designers' interface realization agents
designers

1 - choice of efficiency parameters; 2 - definition of input data for a choice of efficiency calculation models, specification of the requirements to parameters; 3 - construction (choice) of models for definition of efficiency parameters; 4 - formation of set of allowable variants; 5 - evaluation of variants on the chosen parameters; 6 - choice of variant with greatest efficiency; 7- acceptances of the decision about decrease (specification) of the requirements or updating telecommunication system structure. D - design data and documents.

Fig.4. The generalized scheme of efficiency evaluation and decision variant choice

which work out and accept decisions, choice realization purpose, criteria of purpose achievement degree evaluation, scales of measurement by criteria of purpose achievement degree evaluation, initial alternatives given for generation, alternatives generation rules, restrictions for alternatives choice, allowable alternatives, correspondence of allowable alternatives set. to criteria estimation of their outcomes, system of preferences, alternatives selection rules and chosen alternatives.

Within the framework of *integrated decision support model* the formal representation of computer-aided design as hierarchy of decision support tasks is given. The design is given as set of iterations, each of which consists of consecutive processes represented on a tree of telecommunication system description and allowing to carry out the integrated solving of synthesis and analysis tasks.

The models of documents are submitted by a set of document structures represented on three levels: *conceptual* (describing document semantics and reflecting contents and character of information included in document), *logic* (specifying internal structure of document and determining its syntactic construction as a composition of separate data elements) and *representative* (describing document appearance). This levels are described by means of contextual-free grammatics, and a set of operations on documents. *Models of document formation and processing* determine interrelation of document groups and document circulation scheme. They are based on the "workflow" concept and represent a interrelation of documents and separate design and planning tasks [Kotenko et al., 1995e].

4.2 Computing models of automated design and planning processes

The computing models describe used in SDPTS methods of data and knowledge representation and manipulation, and also determine various aspects of realization of the design and development planning of telecommunication systems (iterating, uncertainty, temporary aspect, parallelism and overlapping in time of various processes, etc.). They are submitted by family of models: (1) deductive and non-deductive reasoning models, (2) models of reasoning under the incomplete and contradictory information, (3) case-based reasoning models, (4) constraint-directed reasoning models, and also (5) agents' coordination, cooperation and collaboration models.

The deductive reasoning models are developed by means of modal logics with truth maintenance, realizing the concept of possible worlds in logic S4. *The non-deductive reasoning models* set design by bidirectional process: (1) by generation of decision variant on the basis of design knowledge and telecommunication systems specifications by means of abduction; (2) checks of conformity of decision variant to the specifications by means of deduction. The incomplete knowledge base manipulation mechanisms are determined by the use of non-monotonic reasoning methods, in particular of circumscription methods, realizing elimination of contradictions and knowledge updating at the expense of definition of the situations, which have caused the contradiction, as exceptions and their evident determination as new situations. The given models are based on the scheme submitted, for instance, in [Ohsuga, 1989].

The developed *models of reasoning under the incomplete and contradictory information* are founded on concept of deductive system and integration of quantitative and qualitative methods. The offered approach is based (1) on structuring of existing methods of the incomplete and inconsistent information processing; (2) on analysis of knowledge uncertainty kinds and use of uncertain information combined processing models; (3) on embedding of models of incomplete and inconsistent information processing in design process models; (4) on defining of techniques ensuring a combined processing of different kinds of uncertainty and admitting adjustment on various approaches [Kotenko, 1995c]. The scheme of utilization of the incomplete and inconsistent information processing methods in SDPTS determines ways of representation of facts, uncertain values, operational knowledge, ways of use of quantifiers, modifiers, modalities, ways of certainty factors interpretation, mathematical apparatus for operating with uncertainty, ways and methods of reasoning, actions in a conflict situation. *The combined processing models* are given by means of attributes of telecommunication system elements, attribute definition ranges, an evaluation function of definiteness of the attributes and rules meanings, identifiers of alternative decision variants, functions and predicates describing telecommunication systems and design processes, rules and precedents reflecting ways of design realization, a component of a choice and adjustment of approaches to a reasoning with uncertainty. According to *model of design under the incomplete and inconsistent information* the design is carried out on the basis of uncertain purposes in space of uncertain conditions produced according to combined processing model.

The offered *models of computer-aided design by means of case-based reasoning* use designers' non-formalized knowledge and combine methods of derivational analogy and expert reasoning. These models include (a) models of precedents, (b) models of reasoning at separate levels of design process hierarchy and (c) integrated models of reasoning [Kotenko, 1998b]. The precedents are given as the interconnected dynamic data structures (corresponding to lines of design processes of various abstraction levels, conditions of their realization and possible failures), connected by discrimination networks. *The model of reasoning at separate levels of design process hierarchy* are intended for definition, adaptation and fixing of precedents at the given design hierarchy levels. Indexing of precedents is carried out according to the descriptions of the purposes, conditions, constraints, importance meanings and failures by means of discrimination networks. By search of precedents at first the design purposes are compared and descriptions of constraints and failures are analyzed, and then the most important elements of the condition are estimated and the weighed incomplete conformity is applied to a choice of the most suitable precedent. *The integrated models of case-based reasoning* are based on hierarchical design methodology using constraints, indexing and detection of failures. The resulting project is created by means of detailed elaboration of precedents, describing the top levels design processes, on subprocesses and subtasks, while the design process is not reduced to non-expandable precedents of tasks and operations. Use of dynamic data structures and the dynamic controlled by the user backtracking allows: (a) to adapt to expert preferences by changing of soft constraint values, being learned on the basis of fixing explanations; (b) to predict, to find out and to correct incorrect decision variants; (c) automatically to modify incorrect knowledge, contained in precedents caused by information incompleteness, being learned on the basis of experience; (d) to notify the designers on mistakes and to make updating the decision variants.

The models of constraint-directed reasoning are given as structure, which elements are (1) ordered set of choice sets (determining decision variants), (2) assignment for set of choice sets (describing the chosen decisions variants), (3) set of restrictions on choice sets and (4) algorithms of constraint-directed reasoning. The design tasks, reduced to constraint satisfaction tasks, are determined in the terms of choice sets and a constraint set. The decision of a design task is a set of alternatives, each of which is chosen from the appropriate choice set and satisfies the constraints.

Models of the agents' coordination, cooperation and collaboration make (1) models of design actions, (2) models of conflict resolution and "useful" relation utilization, (3) negotiation models and (4) general multi-agent planning models. These models are based, mainly, on results of works [Lander et al., 1991], [Martial, 1992], [Kotenko, 1994a] and [Kotenko et al., 1995a]. *The models of actions* are founded on the use of action representation, basing on events, which supports the obvious description of time, used resources and relations between actions. *Models of conflict resolution and "useful" relation utilization* include the description of multi-agent design plan, models of temporal and resource conflict resolution, and also models of plan modification based on help relation application. Taxonomy of relations between the various agents' plans includes negative relations (conflict for a consumed and non-consumed resource, incompatibility of actions), positive relations (equivalence, inclusion, help) and inquiry relations. The presence of relations between the agents' plans means an opportunity of updating these plans for their coordination purpose. The negotiation process is considered as a structured interaction by message transfer between various agents. *The negotiation models* are given by means of dialogue condition, types of agents' interaction messages, and negotiation protocols (determining in what dialogue condition with what the message type an agent can sent or be received). *The general multi-agent planning models* are submitted by structure, which elements are design environment (determined by set of agents and their opportunities), states of initial agent coordination and stages of plan coordination. The considered models are fixed in the coordination, cooperation and cooperation agents.

5 Formalization of design and planning objects

The formalization of telecommunication systems gives the SDPTS intelligent agents an opportunity to manipulate the knowledge of base principles of telecommunications, organizational and technical construction of telecommunication system elements and fundamental physical laws of their functioning required for design realization [Kotenko et al., 1996].

The following *models of design objects* are developed: (1) conceptual model of telecommunications subject domain intended for the generalized description of telecommunication systems and formation of individual models of their elements; (2) models of structure and parameters of telecommunication systems serving for representation and manipulation of attributes of the structure and parameters of telecommunication system elements; (3) models of telecommunication system functioning processes, reflecting dynamics of changes, occurring at functioning, and the descriptions of sequences of processes affecting telecommunication system elements.

The conceptual model of telecommunications subject domain is determined as structure specifying subject domain objects, their properties, attributes of properties, object functioning processes, relations between objects, states and situations of object functioning. The objects represent essences to which there can correspond as actually design objects as well as factors influencing them.

The models of structure and parameters of telecommunication systems are intended for compact representation of structure and parameters of design objects by set of typical elements and relations between them, and also effective automatic manipulation with their structures and parameters by means of operations of element decomposition, element composition, calculation of values of element parameters, etc. The models are described by the use of attribute grammatics:

M_{SP}=<V_T, V_N, S, P, O, A, R>,

where V_T - terminal symbols (describe primary elements of telecommunication systems); V_N - non-terminal symbols (appropriate to derivative elements); $S \subset V_N$ - initial symbols (describe set of derivative elements, occupying the top levels of hierarchy of telecommunication systems); $V=V_T \cup V_N$, $V_T \cap V_N=\varnothing$; P - rules of transition from one elements to other; $O=\{\&, \vee, \oplus, ||, \otimes\}$ - operations of element combination, where $\&$ designates conjunction, $\vee$ - disjunction, + - excluding "or", || - "multiform" and $\otimes$ - "alternativity" of elements. Thus the conjunction ($\&$) corresponds to consecutive connection of elements, disjunction ($\vee$) - parallel connection, excluding "or" ($\oplus$) - an opportunities of realization of elements by means of a choice of one from several allowable elements, "multiform" (||) - connection of alternative variants of element structure representation, alternativity ($\otimes$) - definition of various variants of element construction; A - attributes describing the element parameters; R: $V \to A$ - rules of definition of values of element parameters. The example of design element structure, represented in fig.5 by attribute grammatics M_{SP} demonstrates the telecommunication network design generation scheme.

The models of telecommunication system functioning processes are given by means of the descriptions of situations, processes and histories. The *situations* fix a set of the involved elements of telecommunication systems and their states. The *processes* set changes occurring at functioning of telecommunication systems, determining integrity restrictions on attributes of elements and their derivative. The *histories* of telecommunication system functioning define sequences of processes

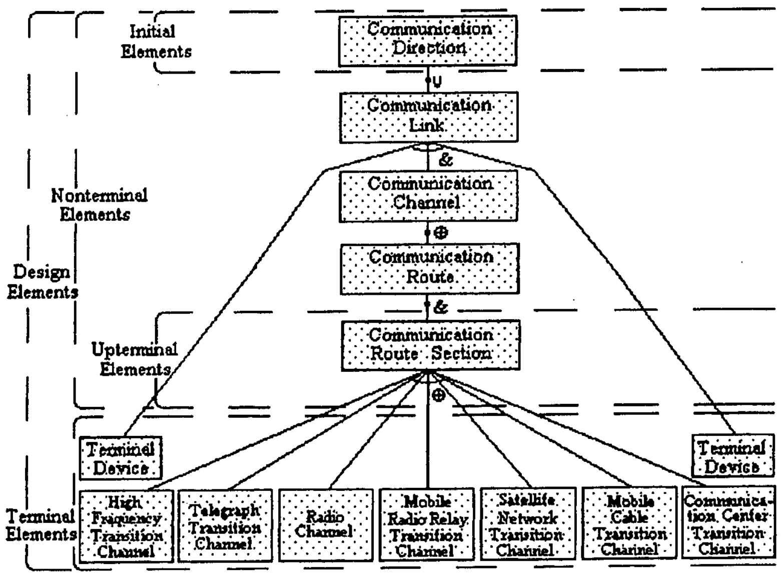

Initial Elements
Communication Direction
∪
Communication Link
&
Communication Channel
⊕
Communication Route
&
Communication Route Section
⊕
Nonterminal Elements
Design Elements
Upterminal Elements
Terminal Device
Terminal Device
Terminal Elements
High Fraquency Transition Channel
Telegraph Transition Channel
Radio Channel
Mobile Radio Relay Transition Channel
Satellite Network Transition Channel
Mobile Cable Transition Channel
Communication Center Transition Channel

Fig.5. Example of design element structure

affecting telecommunication system elements. The models are based on mechanisms of qualitative reasoning [Forbus, 1984] and their basic features are application of qualitative meanings of attributes, representation of subject domain "laws" as constraints on meanings of parameters and contraction of analysis of telecommunication system functioning to a constraint feasibility task.

6 Models and algorithms of intelligent design agents' functioning

The offered models and algorithms of design agents' functioning serve for realization of SDPTS design environment and are considered below.

6.1 Models and algorithms of base task solver functioning

These models and algorithms are developed on the basis of ends and means analysis method and an intelligent casual-commitment strategy [Veloso, 1994].
The base task solver is a functionally complete nonlinear scheduler, since it can interleave goals and subgoals on any search space depth, and can use the given or automatically acquired control knowledge in all the choice points of task solving. Base task solver is based on use of the entered knowledge representation language: the operators are represented as the scripts or reasoning rules set by conditions and effects.

6.2 Models and algorithms of case-based reasoning agents' functioning

These models and the algorithms include the generalized model of case-based reasoning agents, and also models and algorithms of these agents' functioning: automatic generation of precedents, formation of precedent base, search of precedents and fulfillment of precedents by analogy. The given models and algorithms are based on the entered computing models of the case-based computer-aided design [Kotenko, 1998b].

6.3 Models and algorithms of constraint-directed reasoning agents' functioning

These models and algorithms are determined by the use of constraints given in the terms of Boolean expressions and described in the form of dependency network, including control information (for distribution of constraints) and information on dependencies (for an explanation and realization controlled by dependence backtracking) [Croker et al., 1993]. The models of functioning of the constraint-directed reasoning agents are based on use of two components: task solver (being expansion of SDPTS base task solver), which operates with knowledge of telecommunication system subject domain and contains information on design task variables and preference ordering, and the reasoning mechanism, controlled by constraints, which traces a state of constraints and directs task solver. Features of these model are maintenance of constraint distribution by giving additional semantics to nodes and connections in a constraint network; use of truth maintenance; maintenance of incremental modification of decisions at change of task specification.

6.4 Models of expert criticism agents

The offered models are intended for intensification of designers' creative activity, protection them from mistakes fulfillment, and decision variant modifications. These models include: (1) general model of the mistakes fulfillment and criticism implementation and (2) actually expert criticism agents' model. The expert criticism agents receive on an input the description of a task and (or) the decision variant offered by the designers or other software agents, and as result generate criticism of reasoning and used knowledge. In a basis of the given class of the agents the ideas were fixed stated, for instance, in [Silverman et al., 1992].
The general model of mistakes fulfillment and criticism implementation allows to predict a place and reason of design mistakes. It is given as structure, which elements are: (a) categories of knowledge (irrelevant, correct, passed and absent), (b) knowledge mistakes (as a result of irrelevant knowledge use, relevant knowledge omission, and necessary knowledge absence), (c) reasons of mistakes, (d) information, displayed at criticism, and also (e) transformations of knowledge category, mistakes and reasons of their fulfillment to criticism information.
The expert criticism agents' model is given as structure with the following elements: (a) influence rendering agents (for realization of "positive" connection with the designer with the purposes of possible mistakes detour); (b) deviation detection agents (for revealing the mistakes and "negative" feedback with the designer); (c) direction agents (assisting to remove mistakes and deviations at designing, specifying possible directions of actions); (d) explanation agents (for improvement of perception and understanding by the designers of criticism realized); (e) agents of criticism strengthening, including recurrence agents (allowing by various ways to execute the same criticism strategy); persuasion agents (describing reasons of the mistake and result of its fulfillment); argumentation agents (allowing to explain

importance of criticism to show the alternative points of view); (f) the relations between criticism agents and types of mistakes (for linkage of criticism mechanisms in a decision network).

By the combined use of these agents (fig.6) the criticism adequate for the current design task, conditions and designers is realized.

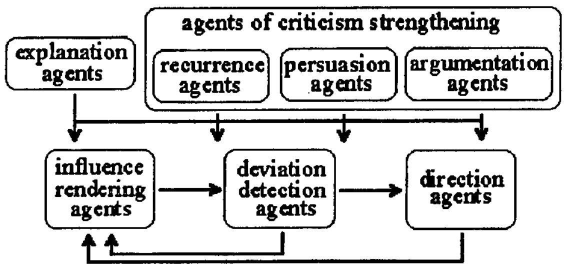


Fig.6. Scheme of use of criticism agents

6.5 Models of designer's cooperative work support agents

The developed models are based on the principles, methods and strategies of intelligent cooperative design and complete conflict resolution between offered decision variants ([Lander et al., 1991], [Martial, 1992], [Kotenko, 1994a], [Kotenko et al., 1995a] and [Kotenko, 1995d]).

As necessary for realization in SDPTS the following *aspects of conflict management* are allocated: (1) conflict avoidance, (2) conflict detection, (3) conflict resolution, (3) decision explanation and updating.

The *conflict avoidance* is supported by using least-commitment and constraint propagation approach, since the design agent is not forced to arbitrary choose from a set of acceptable alternatives simply to make a definite commitment and the choice are carried out by using actual constraints.

Two *approaches for conflict detection* are offered: (1) *qualitative*, basing on hierarchy of possible conflict types, where conflicts are shown through set of the broken restrictions; (2) *quantitative*, based on an estimation of local and global parameters describing compatibility of alternatives and satisfaction of restrictions, where conflicts are shown, when the required meanings of parameters of the alternatives compatibility and satisfaction of restrictions are not satisfied.

When conflict is detected one of the designer's cooperative work support agents must execute a protocol for *conflict resolution* process. To make this the agent should know the conflict situation information, the set of conflict resolution strategies (tab.1), extended by means of set of rules and heuristics (tab.2, tab.3), and scheme of a strategy choice on the basis of conflict symptoms.

Tab.1. Conflict resolution strategies

№	Strategy	Comments
1	Generate Random Alternatives	If multiple solutions exist or can be generated easily, then choose the proposals rated equally or slightly lower
2	Find Compromise	Find the solution that is within the acceptable interval using agent's relaxation of variable values
3	Generate Constrained Alternatives	Generate new alternatives based on constraints received from inflexible agent or using other agent's partial solution
4	Generate Goal Alternatives	Generate proposals by looking for alternate goal enlargements. Some goals can be relaxed or relinquished
5	Find Precedent Solution	Find a previous solution succeeded in resolving a similar conflict situation
6	Revise and Merge Goals	Build a new goal structure that incorporates goals of all agents involved in conflict and generate a solution guided by these structure

The *action explanation* is realized by the metadesign mechanisms, which in the elementary case to each rule or heuristics correspond the explaining statement.

Tab.2. General conflict resolution expertise heuristics

№	*Domain-independent heuristics*
1	If design agents have alternative contradictory plans for achieving their design goals, then find an alternative way of achieving their design goals that is not contradictory
2	If something has a negative influence, then put obstacles in its way
3	If something has a negative influence on the object, then locate the object to the place where this influence is not a concern
...	

Tab.3. General conflict resolution expertise heuristics

№	*Application-specific heuristics*
1	If the fixed area network can't guarantee desired quality indexes, then the mobile straight link network must be planned
2	If area network communication center is developed in considerable distance from control center, then link between them is realized by means of radio relay stations
...	

6.6 Model of document forming and processing agents

These agents are intended for complex automation of document forming and processing. They are developed as a result of a specification of the document models and document forming and processing models given at the description of prescriptive design models. They are constructed according to allocation of set of functions of document development of various level: design as a whole, design of separate subsystems, realization of technological operations and their elements [Kotenko et al., 1995e].

6.7 Models of specifications and design version forming agents

The offered models are intended for representation in the SDPTS information repositories of generated decision variants evolution, including forming and analysis of alternative decision variants, realization of their consecutive improvements, coordination of various aspects of telecommunication system elements, "assembly" these elements from elements, included in their structure, etc.

These models include the generalized model of decision variant specifications forming and the version management model. The first model is submitted as structure, which elements are the specifications such as design objects, configuration specifications, specification at a level of exemplars and specification generation mechanisms. The second model sets design object versions, their configurations, version histories, working areas and operation on versions.

6.8 Models and algorithms of guaranteed time maintenance and design realization agents' functioning

These models and algorithms are developed for maintenance of the timeliness, design validity and controllability requirements and are intended for design technology management and adaptive forming of an optimum (rational) sequence of design and use for each separate design process (task, procedure) of the most acceptable models and algorithms allowing to obtain the decision for allocated time with required (or maximum achievable) validity. They include (1) common model of temporary restriction satisfaction, (2) common algorithm and mechanisms of the design and temporary restriction satisfaction management, (3) models and algorithms of real time design realization. The given models are based on works on deliberative real-time artificial intelligence [Garvey et al., 1994].

The common model of temporary restriction satisfaction is submitted as structure describing (1) design conditions, (2) temporary restrictions, (3) computing resources, (4) models of the design processes, tasks and procedures, (5) structure of the SDPTS agents (allowing to realize these models by means of procedures having various computing complexity, accuracy, definiteness and completeness), (6) agent's profiles (expected temporary expenses at their use), and also (7) rules of agent choice by the use of "flexible" calculations, realizing incremental decision improvements, and contract calculations which are forming the design plan and the inclusion in it of necessary approximations.

The common algorithm of design and temporary restriction satisfaction management includes: formation (updating) of design models - generation of chains of agents and their actions (generation of approximations); forecasting of solution time and parameters of decision variant quality; the revealing, whether design will be executed for the allocated time with required quality (recognition of approximation necessity); design fulfillment according to the model (realization of the chosen approximation by the "rigid" algorithms, rough search strategy, approximate data and knowledge); dynamic forecasting and observation of design parameters (including design time); identification of design condition and formation of design approximations, processes, tasks and procedures.

The models and algorithms of real time design realization are based on transformations of three kinds: (1) transformations of design plan (model) from a stored format (as precedent or script) in operational representation (as records of actions) used for design realization; (2) transformations of a sequence of action records in a SDPTS fulfillment line; (3) transformations of a fulfillment line in representation of precedent (or script) for fixing in SDPTS information repository.

6.9 Models of teaching and work accompaniment agents

The teaching and work accompaniment agents are intended for teaching to the basic concepts, design knowledge and facts, skills of an evaluation of decision consequences, tracking designers' work and rendering help to them. The offered models and algorithms are based on architecture of intelligent help systems ([Breuker et al., 1987], [Erlandsen et al., 1987]). These models and algorithms are include: (1) model of teaching and work accompaniment, (2) models and algorithms of teaching management (based on a plan generation method controlled by the goals), (3) models and algorithms of recognition and forecasting of designers' activity (including interpretation of inquiries and actions), (4) learner models, (5) models of control question generation, (6) models of soluble tasks, (7) models of estimation formation and mistake diagnostics, (8) models of generation of communication conditions and of formation of teaching task conditions, (9) models of changes in telecommunications, (10) models of task solving, (11) models of knowledge acquisition [Kotenko et al., 1994b].

6.10 Models of information security agents

The models of information security agents allow to reflect all design aspects that important from the point of view of protection against the unauthorized access. The basic information security agents are (1) the access differentiation agents, (2) the unauthorized access detection agents and (3) the authentication agents.

The model of the access differentiation agents is given as models of (1) mandatory and (2) discretionary access differentiation rules implementation, reflecting accordingly processes of the private information flows management (which are not admitting of outflow on unauthorized access channels), and processes of designers' and software agents' access to information resources according to their functional role. *The model of realization of mandatory access differentiation rules* is submitted by dynamic, marked, directed graph, which nodes make the usual and trusted processes, the directed edges set the messages between processes, and the labels serve for representation of information privacy levels. The relations are entered that fixing assignment of levels to usual processes and the authorized design messages, and also definition of a maximum level of trusted processes. On the basis of specified elements and sets of design events the mandatory access differentiation rules are offered. *The model of realization of discretionary access differentiation rules* includes sets of the subjects (active design elements) and objects (SDPTS resources), their classes, categories of subjects and objects access, binary predicate symbols, compared with a kind of the subjects to objects messages, and discretionary access differentiation rules represented as productions, which left and right parts are given by predicate formulas. The unauthorized access is treated as discrepancy between a real object condition and a condition determined by discretionary rules.

The model of the non-authorized access detection agents describes processes of search and elimination of the non-documentary functions and mistakes in SDPTS agents' structure, and also definition of deviations of the designers' and software agents' actions from authorized order. This model is given by sets of control records (representing agents' messages and generated at functioning or testing of SDPTS agents), profiles of normal behaviour (describing subjects' behaviour in relation to objects using statistical parameters), records of abnormal behaviour (created at detection of deviations from some profile of normal behaviour), reaction rules (being the description of conditions and actions which are carried out by analysis results) and unauthorized access detection rules (allowing to determine an approximate estimation of belonging of message to unauthorized access channel).

The model of the authentication agents sets distribution of safe channels of message exchange between design processes, sanction of initialization of processes only to given set of active components and maintenance of documentary confirmation of information exchange. It is given by sets of process responsibility carriers, groups of responsibility carriers, messages, the base relations between the responsibility carriers agents and reasoning rules for obtaining of the new relations. The SDPTS functioning is submitted by generation of messages by responsibility carriers. For an establishment of relations between responsibility carriers the two reasoning rules are used: (1) responsibility transmission and (2) authority delegation. The set of responsibility carriers is given with a responsibility carrier name and open key from pair of keys of responsibility carriers for asymmetric enciphering algorithm. The authentication processes are described by creation and check of the certificates representing right parts of reasoning rules. The message exchange is accompanied by certificate exchange, with which help for each process on various information processing units by application of reasoning rules to the certificates the responsibility carriers are determined.

7 Implementation and efficiency evaluation

The following *SDPTS agents' prototypes* are realized: (1) cooperative work support agent (CWSA), (2) document development agent (DDA), (3) hypermedia information support agent (HISA), (4) case-based reasoning agent (CBRA) and (5) separate agents for design of telecommunication subsystems.

For agents' realization the Smalltalk, C++ and Delphi were used. We've realized more then 120 classes of objects. The part of class hierarchy realized in C++ is depicted in fig.7.

The scheme of telecommunication system design and the responsibility spheres of agents' realized are depicted in fig.8.

The basic aspects of these agents' realization are:

(1) the *CWSA* realization is based on the developed models of coordination, cooperation and collaboration agents and designer's cooperative work support agents. The architecture of agents' interaction is based on common blackboard, intended for representation of the decisions, negotiation and coordination variants;

(2) *DDA* has the following features agreed to the SDPTS construction conception: the document formation reflects process of cooperative design; as a result of their functioning together with the designers in information repository the necessary documents are collected; the method of work "on precedents" is realized; the means of various document versions support are used; the automated document workflow is supplied;

(3) *HISA* realizes a two-level model of hypermedia-system and consists from multimedia database organized as open, changeable network of elements, control system ensuring performance of hypermedia-system base functions and designers' interface;

(4) the realization *CBRA* is based on the case-based reasoning model developed. At reception of a new task CBRA finds suitable precedents, addressing to rule base and precedent base. The decision variant is formed by means of repeated fulfillment of decision process fixed in precedents for a new situation.

In view of complexity of the telecommunication system design processes automation the complete construction of SDPTS remains in the long term.

So the SDPTS construction decisions were tested by means of analytical and imitating models, and also as a result of experiments with the software agents' community realized. *The estimation* was carried out for cases of realization of traditional design technology and various variants of the offered technology of computer-aided design and development planning of telecommunication systems. For estimation of efficiency of separate design processes and basic SDPTS agents the processes of formation of documents and preparation of decision variants and also the DDA,

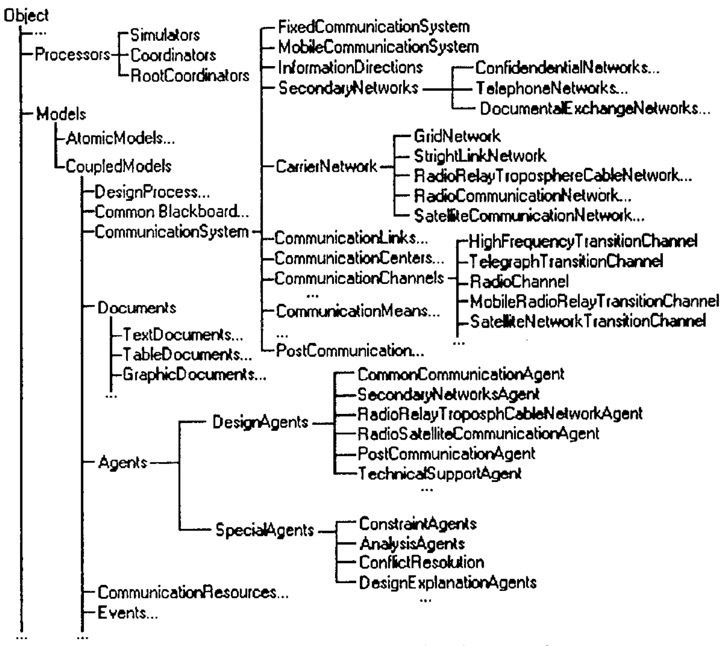


Fig.7. Classes of Objects implemented

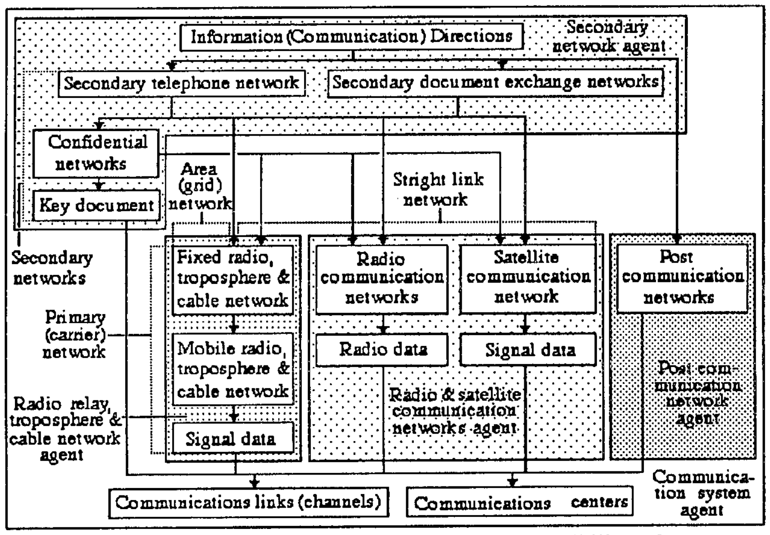

Information (Communication) Directions
Secondary network agent
Secondary telephone network
Secondary document exchange networks
Confidential networks
Key document
Area (grid) network
Stright link network
Secondary networks
Primary (carrier) network
Radio relay, troposphere & cable network agent
Fixed radio, troposphere & cable network
Mobile radio, troposphere & cable network
Signal data
Radio communication networks
Satellite communication network
Radio data
Signal data
Radio & satellite communication networks agent
Post communication networks
Post communication network agent
Communications links (channels)
Communications centers
Communication system agent

Fig.8. Design object generation and agents' responsibility sphere

HISA and CBRA agents were chosen.

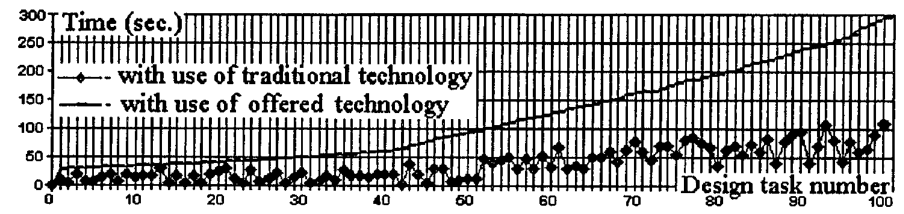


Fig.9. Execution time of design tasks

Proceeding from the carried out analysis, a conclusion about achievement of required meanings and significant gain on the basic parameters of essential design properties has been made at realization of offered technology in comparison with existing:

1) on timeliness: effect of reduction of critical way time for various design conditions - *1.2÷2.2* times (for separate stages) and *1.2÷3.5* times (for all design cycle). Thus on each of stages the additional time, transferred for the benefit of decision development processes by the designers, - *30÷70%*. As an example in a fig.9 for each task from a set of realized design tasks, ordered on complexity, the evaluations of execution time with use of traditional and offered technology are submitted;

2) on validity: effect on an integrated parameter of the decision validity and accuracy expressed by decrease of entropy of considered decision variants number achievable at designing - *20÷40%*;

3) on controllability: the offered technology realization allows to increase parameters controllability at the expense of an opportunity of a variation for the allocated time of number of iterations and variants researched. The increase of meanings of these parameters conducts to validity growth, and achievement of the given meanings for smaller time - to increase of timeliness at allowable validity. A prize on number of iterations - *1.5÷6.4* (times), prize by quantity of taken into account variants - *1.1÷14.2* (times);

4) on laboriousness: a gain on factor of designers' loading - *1.1÷1.4* (times), a gain on factor of designers' intensity - *1.4÷2.7* (times);

5) on security: the allowable unauthorized access probability is provided at minimization of information protection time on critical to time functioning sites, or at reduction of volume of the resources spent on unauthorized access detection.

Bibliography

[Brenner et al., 1998] *Brenner W., Zarnekow R., Wittig H.* Intelligent Software Agents. Foundations and Applications. Springer-Verlag, 1998.

[Breuker et al., 1987] *Breuker J.A., Winkels R.G.F., Sandberg J.A.C.* A shell for intelligent help systems // IJCAI-87. Proceedings of the 10-th International Joint Conference on Artificial Intelligence, 1987.

[Chandrasekaran, 1986] *Chandrasekaran B.* Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design // IEEE Expert, 1986. Vol.1. No. 3.

[Coyne, 1990] *Coyne R.* Logic of Design Actions // Knowledge-Based Systems, 1990. Vol.3. No. 4.

[Croker et al., 1993] *Croker A.E., Dhar V.* A Knowledge Representation for Constraint Satisfaction Problems // IEEE Transactions on Knowledge and Data Engineering, 1993. Vol.5. No. 5.

[Durfee et al., 1989] *Durfee E.H., Lesser V.R., Corkill D.D.* Trends in Cooperative Distributed Problem Solving // IEEE Transactions on Knowledge and Data Engineering, 1989. Vol.1. No. 1.

[Erlandsen et al., 1987] *Erlandsen J. Holm J.* Intelligent help systems // Information and Software Technology, 1987. Vol.29. No. 3.

[Forbus, 1984] *Forbus K.* Qualitative process theory // Artificial Intelligence, 1984. Vol.24.

[Garvey et al., 1994] *Garvey A., Lesser V.* A Survey of Research in Deliberative Real-Time Artificial Intelligence // Journal of Real-Time Systems, 1994. Vol.6. No. 3. P.317-347.

[Gorodetski, 1996] *Gorodetski V.I.* Multi-Agent Systems: State of the Art and Perspectives // Artificial Intelligence News, No. 1, 1996 (in Russian)

[Kleinfeldt et al., 1994] *Kleinfeldt S.* et al. Design Methodology Management // Proceedings of the IEEE, 1994. Vol.82. No. 2.

[Kotenko, 1994a] *Kotenko I.V.* Conflict Resolution in Computer-Supported Cooperative Design // Lecture Notes in Computer Science. Berlin, etc.: Springer Verlag, 1994, Vol.876.

[Kotenko et al., 1994b] *Kotenko I.V., Kopjlov A.I.* Using AI planning techniques for formation of teaching and work accompaniment strategies in intelligent decision support systems // East-West International Conference on Computer Technologies in Education. EW-ED'94. Proceedings. Part 2. Crimea. 1994.

[Kotenko et al., 1994c] *Kotenko I.V., Golikov V.V, Janchenko I.P.* Computer-Aided Technologies of the Communication Network Design and Development Planning // International Information Forum. III International Conference on Informational Networks and Systems. ISINAS-94. St.Petersburg. 1994.

[Kotenko et al., 1995a] *Kotenko I.V., Krechman D.L.* Computer-Aided Negotiation Support in Hypermedia Multi-Agent Systems // CMC'95. International Conference on Cooperative Multimodal Communication, Theory and Applications. Proceedings. Eindhoven. The Netherlands. 1995.

[Kotenko, 1995b] *Kotenko I.V.* A Framework for Intelligent Group Decision Support // 3rd European Conference on Information Systems. ECIS'95. Proceedings. Athens. Greece. 1995.

[Kotenko, 1995c] *Kotenko I.V.* Uncertainty Management in Intelligent Decision Support Systems // International Joint Conference of CFSA/IFIS/SOFT'95 on Fuzzy Theory and Applications. Taipei, Taiwan, 1995.

[Kotenko, 1995d] *Kotenko I.V.* Intelligent Computer-Aided Design: Strategies and Models for Collaborative Work Support // KDS-95. International Conference "Knowledge-Dialogue-Decision ". Proceedings. Yalta, 1995. Vol.2.

[Kotenko et al., 1995e] *Kotenko I.V., Janchenko I.P., Bogovik A.V.* Computer-aided development of communication documents. SPb: MSA, 1995 (in Russian).

[Kotenko et al., 1996] *Kotenko I.V., Rabov G.A., Saenko I.B.* Intelligent systems for management of telecommunications. SPb: MSA, 1996 (in Russian).

[Kotenko, 1998a] *Kotenko I.V.* The theory and practice of computer-aided systems construction for telecommunications planning on the basis of new information technologies. SPb: MSA, 1998 (in Russian).

[Kotenko, 1998b] *Kotenko I.V.* Models of Case-Based Reasoning for Realization of Intelligent Systems // CAI-98. Sixth National Conference with International Participations "Artificial intelligence - 98". Proceedings. Part 1. Pushchino, Russia, 1998 (in Russian).

[Lander et al., 1991] *Lander S.E., Lesser V.R., Connel M.E.* Knowledge-based Conflict Resolution for Cooperation among Expert Agents // Lecture Notes in Computer Science, 1991. Vol.492.

[Martial, 1992] *Martial F.* Coordinating Plans of Autonomous Agents // Lecture Notes in Computer Science, 1992. Vol.610.

[Ohsuga, 1989] *Ohsuga S.* Toward intelligent CAD systems // Computer-aided design, 1989. Vol.21. No. 5.

[Silverman et al., 1992] *Silverman B.G., Mezher T.M.* Expert Critics in Engineering Design: Lessons Learned and Research Needs // AI Magazine, 1992. Vol.13. No. 1.

[Tomiyama, 1986] *Tomiyama T., Yoshikawa H.* Extended General Design Theory // Rep.CS / Centrum voor Wiskunde en informat. Computer Science, CS 8604. Amsterdam: Stichting math. Centrum, 1986.

[Veloso, 1994] *Veloso M.V.* Prodigy/Analogy: Analogical Reasoning in General Problem Solving // Lecture Notes in Artificial Intelligence, 1994. Vol.837.

[Wooldridge et al., 1995] *Wooldridge M., Jennings N.* Agent Theories, Architectures, and Languages: A Survey // Proceedings of the Workshop on Agents Theories. Lecture Notes in Artificial Intelligence, Vol.890, 1995.

[Wooldridge et al., 1998] *Wooldridge M., Jennings N.* The Pittfalls of Agent-Oriented Development // Proceedings of the Second Conference on Autonomous Agents (Agents'98), 1998.

MULTI-AGENT SYSTEM TO MODEL THE FISHBANKS GAME PROCESS

Jaroslaw Kozlak[1], Yves Demazeau[2], Francois Bousquet[3]

[1] *Katedra Informatyki AGH, Al. Mickiewicza 30, 30-059 Krakow, Poland, E-mail: kozlak@agh.edu.pl*

[2] *Laboratoire Leibniz-IMAG, 46 avenue Felix Viallet,38031 Grenoble Cedex, France, E-mail: Yves.Demazeau@imag.fr*

[3] *CIRAD, Campus de Baillarguet 34032, Montpellier Cedex 1, France, E-mail: bousquet@cirad.fr*

Abstract

The article presents decentralized multi-agent system for the FishBanks game simulation. The participants of the game are the fishing companies aiming at generating maximum profits on fishing while sticking to their assigned limits of fish catching thus avoiding excessive overexploitation of the fish-banks. The system allows the participation of both computer-simulated players as well as real players. The system enables to research the problems of reaching agreement in negotiations, maintaining the common renewable resources on the appropriate level (the problem of "the tragedy of commons") and provides the satisfactory functioning of the system regardless of any change in its conditions of operation (the problem of maintaining functional integrity which we are working on).

Keywords: *renewable resources, Fish Banks game, multi-agent simulation, maintenance of the functional integrity*

1. Introduction

FishBanks system is based on the Fish Banks game designed for teaching effective co-operation in using natural resources ([Meadows 93a], [Meadows 93b]). It allows for the analysis of the phenomena which are typical for the whole class of similar problems such as soil erosion, fossil fuel depletion, deforestation, ground water pollution, over-fishing, etc. Several teams of players - fishing companies are the participants of the game. Each company aims at collecting maximum assets expressed by the number of ships and the amount of money deposited at a bank account. The money is generated on fish catching at fish banks - the bigger the catch is, the higher initial profits are; however, excessive fish catching may result in overexploitation of the fish banks resources. The aim of the play is to teach and research the negotiation techniques applied by the company (players' teams) in order to avoid overexploitation of the resources.

The following basic goals were kept in mind while we were creating the computer implementation of the game:

- simulation of the FishBanks game with the artificial players (negotiations as the method of effective application of the renewable resources in the system);
- game with human participation (researching human behaviour and teaching effective strategies in negotiating exploitation of the renewable common resources);
- analysis of the maintenance of the functional integrity of the system (impact that the state of the resources has on maintaining the functional integrity of the system).

The nature of the multi-agent systems (they are composed of the autonomous, intelligent elements) is very suitable for carrying out such experiments.

There is a number of multi-agent systems which deal with the problems of the exploitation of the common renewable resources and they undertake both the analysis of the general problems ([Bousquet96], [Antona98], [Rouchier98]) as well as specific situations when we deal with the problem of the common renewable resources e.g. the analysis of the community of hunters and farmers [Proton97].

2. Description of the "tragedy of commons"

"Tragedy of commons" appears when several users have a free access to the resources. There is always the danger that exploitation of the common resources may result in their overexploitation called "the tragedy of commons". Let's assume that the warehouse of resources is able to offer resources and reproduce them at a certain speed of their consumption by the users. Excessive speed of the consumption may result in overexploitation of the resources. The problem was described by Hardin [Hardin68]. He based on the works of the mathematician William Forster Loyd of 1833 in which Loyd had analyzed utilization of the common pastures by the sheep farmers in England. The pastures constitute common property - they are available for many sheep farmers at a time. Each farmer is interested in having the biggest herd possible. Only a limited number of sheep can graze on the meadows. Problems emerge when the number of sheep reaches the maximum limit permissible for the pasture. On one hand the sheep farmers should not increase their herds any more as it leads to devastation of the pasture and unables to maintain their herds in the future. On the other hand, all the farmers share the inconvenience of increasing the herd in number while profit of this is attached to the one who has increased the herd. Thus rationally reasoning farmers will be increasing their herds.

A similar problem emerges in many situations when the operational effectiveness of user depends on the level of their utilization of the renewable common resources. It concerns fish and whales catching, environmental pollution, preservation of the tropical forests, urban transport problems, etc. The problem "the tragedy of commons" always emerges when the user (not necessary a human being) acts in a rational way - therefore it also relates to the multi-agent system. The problem may emerge in the computer systems which resources are common for many processes - utilization of time resources of the processor, memory, capacity of the message channels, etc. It may additionally include real resources management if they are managed by computer and not by human being. The problem "the tragedy of commons" was presented from the point of view of the multi-agent systems in the work of R.M.Turner [Turner91]. Certain methods were proposed in order to avoid "the tragedy of commons" both for the societies made of rational individuals (mainly human communities) as well as specifically for the multi-agent computer systems [Turner91] :

- **multiagent planning** approaches. One of the agents is elected as the planner and he makes decisions concerning availability of the resources ([Cammarata83], [Georgeff84]). The disadvantage of the approach is that the planning agent needs to posses extensive knowledge of the other agents. Otherwise he will not be able to make assign optimum limits.
- **partial-global planning** [Durfee87]. The agents exchange information on their objectives and plans and a common plan is created. It may be a difficult task to develop such plan for individual cases. Moreover, it is only possible for a certain class of systems e.g. FA/C systems with co-operating agents. If the agents' objectives contradict drastically, it may be impossible to work out such a plan.
- **voluntary measures and conventions**. Voluntary measures are not sure means of problem solving ("free rider problem" - e.g. a participant is not willing to face inconveniences in order to maintain the state of the resources as he may assume that the others will do it for him). Only when the designer has full control over the resources the convention may work.
- **monopolies**. Rights to the resources are given to one agent only. The approach works similarly as the multi-agent planning, however there is no chance here to change the managing agent as he forces his position on the other agents. The resources can be also ruined by the competition for the monopolists position, moreover, it may result in a decrease in the effectiveness of the system providing the decisions made by the monopolists contradict with the objectives of the other agents.
- **privatisation.** It is a simple rule to avoid many cases of "the tragedy of commons", however not all the types of resources can be privatised (the system may not have the information on its actual amount, it may not be able to enforce the rights of ownership on the other agent, indivisibility of the resource).
- **mutual coercion mutually agreed on**. The agents negotiate the right of access to the resources among themselves and jointly undertake certain coercion which prevent

collapse of the system (e.g. introduction of progressive taxation on the amount of the exploited resources). The solution is proposed in [Hardin68] and it is also used in our negotiations in the FishBanks system.

3. Rules of the Fish Banks game

The Fish Banks game is played in succeeding rounds which represent years of the game. Few teams participate in the Fish Banks game. Each team represents one of the fishing companies. Each company aims at collecting maximum assets expressed in the amount of money and ships. The companies are allowed to catch fish in two fishing waters (inshore or deep-sea fishing) or they may keep their ships at the port. Initially it is assumed that the number of fish in deep-sea fishing waters is higher than the number of fish in inshore fishing waters therefore deep-sea fishing is more profitable. Costs of fishing (ships preparation) are higher for the deep-sea fishing and lower for the inshore fishing. The company may also leave their ships at the port and pay even lower rate for their maintenance, however this way it cannot fish. The companies may order new ships to be built as well as they may cross-sell their ships. The ships may be also sold at the auction organized by the game manager. The players can enter negotiations. The subjects of the negotiations are not limited, in fact they typically concern limitations in terms of the fish catching limit in the fishing waters which are endangered with overexploitation. The costs of building a ship, costs of its maintenance and use as well as the cost of sold fish is fixed for the whole game. At the end of the game the value of the ships owned by the companies is estimated. In the course of the game a standard case of "the tragedy of commons" is faced as the fish constitute the common renewable resource.

4. Description of the model

4.1. Introduction.

In the following sub-chapters we describe the model of the FishBanks system: we give the main assumptions, describe the types of resources in the system, the types of the agents in the system, the types of actions performing by the agents, the mechanisms of the reasoning of the agents and the kinds of interactions in the system.

4.2. General assumptions

There are several types of agents in the system (such as company, game manager, ship-builder, fish-seller, fish-bank, weather, visualizer, clock). Each type of agent has the skills to perform its typical actions, also their goals may be similar. The actions performed by the agents may be classified into two groups:

- actions connected with transfer or production of resources;
- actions connected with negotiations to prevent or enforce some actions which belong to the first group.

4.3. Resources

The agents operate using three types of resources: fish, money and ships. Each resource has its specific features.

Fish. Fish are renewable resource. Their number grows in a specific time period Δt (one year of the game) and depends on their actual number and the maximum permissible number:

$$\Delta R = R(t) * R_v * (1-R(t) / R_{Max}) \qquad \text{(Eq. 1)}$$

where:

ΔR -the number of fish born within the period $(t, t+\Delta t)$;

$R(t)$ - the number of current resources;

R_{Max} - the maximum number of resources;

R_v - reproduction co-efficient.

Although fish are actually owned by specific agents (fish bank, company or fish-seller), the fish bank agent must make them available to each agent who is capable of performing appropriate actions, therefore it may be considered as a common resource.

Money. The system views money as renewable resource. Every period (a year of the game) its amount is updated as a result of the interest rate on the account (depending on the negative or positive balance on the account the interest rate may be also negative or positive). The money is also owned by specific agents and there are no limitations as to their use (they always constitute private resources of the individual agents).

Fishing ships. Ships are private resources owned by the agents (company, ship-builder and game manager) and some of them have the capacity to build new ships (ship-builder and game manager). The ships may be also seized i.e. as they are used by the agent for certain actions (e.g. sending to the fish bank) they cannot be used for any other one. Moreover, there is a specific charge for ships storing which are owned by the agents-company).

4.4. Dynamics of the game

The system may remain in several different states during which the agents do some specific tasks. This happens because of the state character of the Fish Banks game and not because of the supposition of the implementation of the system. The agents perform paralelly and their moving on to the others states is preceded by the exchange of the messages used for synchronization. There are 4 regular stages of the game which take place subsequently in each year of the game:

- ships and money collection;
- ordering new ships to be built;
- fish catching;
- resource regeneration.

There are also two special stages of the game to be moved to by the system as requested by the agent:

- the auction;
- negotiations;

4.5. Agents

There are several different types of the agents in the system. Each type has its specified actions to be performed as well as the objectives and strategies of their implementation. The system includes the following types of agents:

- **company** - the agent of such type represents the company participating in the Fish Banks game. He may send ships to the fishing waters to catch fish (*SendShips*), order to build of new ships (*OrderShips*), buy or sell ships (*SellShips*), (*BuyShips*), sell fish (*SellFish*), calculate the state of its account (*CalAccState*), pay charges for the ship storage (*ShipsStorage*). He may also participate in auctions and negotiations.
- **ship-builder** - builds ships and delivers them to the companies which order them (*DeliverShips*).
- **fish-seller** - buys fish (*BuyFish*).
- **game manager** - initiates auctions of ships, he may also execute the agreed rights which come as the result of negotiations.
- **fish-bank** - represents the fishing waters and calculates the current number of fish at the fish bank. The number is subject to fluctuation as a result of catching and regeneration (*FishReg*).
- **weather** - tells the weather on the fishing waters and has impact on the level of catching.

4.6. Actions

The table describes the actions which the agents are capable of performing. The actions are related to the operations on the resources. As a result of an action the resources may be:

- **exploited** - the agents produce the resources, obtain the resources from another agent;
- **used** - the resources are irrevocably spent by the agent;
- **locked** - the resources are not available as long as the action takes place, when it is completed they are available again
- **transferred** - the resources are transferred to another agent

Each action and operation may involve even all kinds of the resources - triples (money, fish, ships), written vertically, are used in their description in the table

Actions	Res. Exploit.	Res. Used	Res. Lock.	Res. Transf.
SendShips (n,m)	0, n*cL(m), 0	n*cC(m), 0, 0	0, 0, n	0, 0, 0
Catching (n)	0, 0, 0	0, 0, 0	0, 0, 0	0, n*cL(n), 0
OrderShips (n)	0, 0, n	0, 0, 0	0, 0, 0	n*pNS, 0, 0
DeliverShips (n)	n*pNS, 0, 0	0, 0, 0	0, 0, 0	0, 0, n
BuyShips (n)	0, 0, n	0, 0, 0	0, 0, 0	n*pS, 0, 0
SellShips (n)	n*pS, 0, 0	0, 0, 0	0, 0, 0	0, 0, n
BuyFish (n)	0, n, 0	0, 0, 0	0, 0, 0	n*pF, 0, 0
SellFish (n)	n*pF, 0, 0	0, 0, 0	0, 0, 0	0, n, 0
ShipsStorage (n)	0, 0, 0	n*cost, 0, 0	0, 0, 0	0, 0, 0
CalAccState (n)	n*add1, 0, 0	N*add2, 0, 0	0, 0, 0	0, 0, 0
FishReg (n,m)	0, ΔR(n,m), 0	0, 0, 0	0, 0, 0	0, 0, 0

Table 1 The meaning of the actions

The meaning of the actions:

- SendShips (n,m) - sending n ships to the fishing waters m;
- Catching (n) - realization of fish catching in the fishing waters with n number of ships;

- OrderShips(n) - an order to build new ships, the transaction takes place only at the beginning of the next year of the game;
- DeliverShip(n) - ship order and delivery;
- BuyShips(n) - purchase of n ships;
- SellShips(n) - sale of n ships;
- BuyFish (n) - purchase of n units of fish;
- SellFish(n) - sale of n units of fish;
- ShipsStorage(n) - charge for owing n number of ships;
- CalAccState(n) - calculation of the new state of the account;
- FishReg(n,m) - regeneration of fish at the fish bank with n number of fish and with the maximum number of fish in the fishing waters which we mark as m.

Values (other than the number of resources) which have impact on the actions are:

- cC(m) - the costs of fishing on m fish bank per one ship;
- cL(m) - the current level of fish catch on m fish bank per one ship;
- pNS - the price of building a new ship;
- pS - the price of one ship sold/purchased;
- pF - sales price of a fish unit;
- cS - the price of the maintenance of the one ship;
- add1 - interest rate on bank deposit (if the level of the account greater then 0);
- add2 - interest rate on debit (if the level of the account smaller then 0);
- R(n,m) the number of new-born fish (described by the formula Eq.1).

4.7. Reasoning

The majority of types of the agents in the system are at the moment reactive agents who react to the events which are happening but don't conduct their own reasoning process. There are two types of agents with more developed decision-making mechanism: game manager and company.

Game manager must make a decision when to start of the auction of ships, their number and the initial price. The current number and the price are specified at random from the numbers included in specific brackets, however the decision about the very beginning of the auction is made when the catch is low and it does not provide the required level (as the level is given and it is expressed by the brackets including the number of fish bought by fish sellers).

More complex decisions have to be made by the **company**. The company's decisions come from its nature which is defined by the following initializing parameters:

- w - parameter describing the strength of the market strategy impact (profit-oriented);
- e - parameter describing the ecological strategy impact (oriented towards maintaining the balance);
- k - parameter describing the achieving information strategy impact (oriented towards collecting information about the system).

Those three parameters above, must sum up to score hundred.

The initial parameters describe, how much is each of these strategies important to the agent.

The agent uses three strategies: gaining fortune, taking care of the balance in the system and achieving the information of the system. The goal function of the agent company is described by:

- W - estimation of the current state of agent from the market strategy position;
- E - estimation of the current state of agent from the ecological position;
- K - estimation of the current state of agent from the archiving information position.

The main goal function (G) of the agent company is described by:

$$G = w * W + e * E + k * K \qquad \text{(Eq. 2)}$$

The agent try to chose the solution which improve its main goal functions. The decisions concern:

- sending ships to fish banks;
- order ship building;
- ships sale/ purchase;
- proposal to start negotiations;
- proposed solutions in negotiations;
- proposal to start auction;
- proposed auction price.

4.8. Interactions

Interaction take place in the system by using messages exchanged by the agents. The messages are created according to the Interaction Language presented in [Demazeau95]. Such messages consist of three segments where each segment transfers a different type of information:

- message language - contains information about the sender and the receiver of the message as well as the identifier of the message;
- application language - contains information typical for the application
- multi-agent language. The language is to present the intention of the sender and his expectations. It contains information about the message protocol used in a given conversation thread and information about the position of the message in this protocol.

The system has three types of the interaction protocols:

- "simple" - inform/perform - used for the action to be performed by agents;
- auction - "escalating bids" - used for the auction of ships;
- negotiations - Sian protocol ([Sian92]) - used for negotiations;

The figures present the complex interaction protocols (of auction and Sian protocol) in use.
The auction is conducted according to the standard protocol ``who gives more". Starting the auction the auctioneer sends the information messages to the participants, The participants initiate the communications threads to remember the state of the auction.

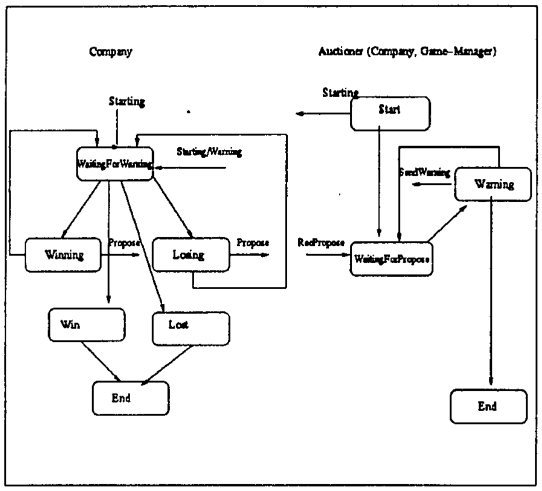


Figure 1. Auction protocol.

Auctioneer (company or game-manager) may be in two main states:

- WaitingForPropose - while waiting for the price proposals;
- Warning - while sending the information about the current state of auction (current winner and his proposition);

The participant (company) may find himself in the following states:

- WaitingForWarning - waiting for the auctioneer's answer with the information about the current state of auction;
- Winning - the participant has proposed the best offer;
- Losing - the other participant has proposed the better offer;
- Win - the participant wins the auction;
- Lost - the participant loses the auction.

The negotiations are based on the Sian protocol [Sian92]. Each proposition is evaluated by the agent, who may accept, reject or ignore it (4th opportunity -- the proposition of the modification is not implemented). Each agent evaluates the course of negotiations and takes the final decision: whether the proposition is accepted or rejected. We assume, so that the proposition would become law it must be accepted by all the agents.

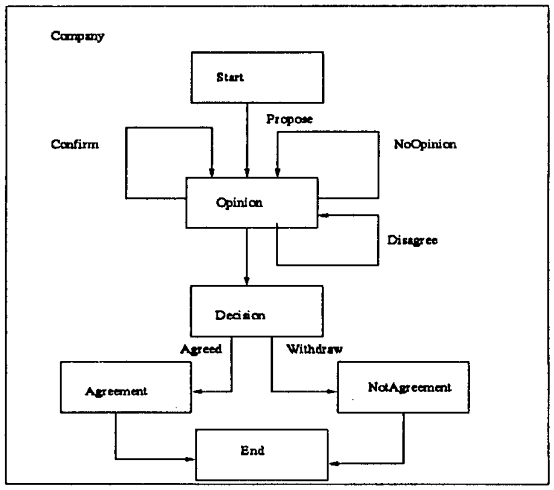


Figure 2. Sian protocol

The negotiated propositions may concern:

- forbidding catching on the particular fish bank;
- accepting catching on the particular fish bank only some quantity of ships for each company;
- accepting catching on the particular fish bank only some percent of ships for each company;
- setting the tax for catching on the fish-bank (game-manager executes it).

4.9. Remarks

Our research concern the problem of the maintenance of the functional integrity by the multi-agent system -- to guarantee that the system will perform its functions independent from the changes in it (linked with the number of the agents in the system, the types of agents or the resources in the system). The FishBanks system makes possible to analyze the influence of the change of the resources to the proper functions of the system. We assume, that system FishBanks works correctly (its functional integrity is maintained) if the quantity of the caught fish in the given time is in the interval of magnitude described by specific numbers.

5. Realization

5.1 Introduction.

In this chapter we present the structure of the FishBank system, the structure of the agent and tools used to implementation of the system.

5.2 Structure of the system.

There are two kinds of processes in the system:

- the process of server - responsible for simulation of game with participation of artificial companies;
- the process of customer - owing to this process a man can participate in the game as one of the fishing companies.

Communication between customer and server is realized by using "remote method invocation" (RMI). The figures present how the processes of server is build (Fig.3)

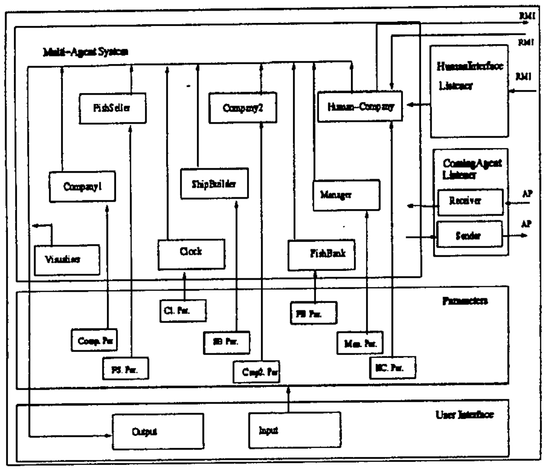


Figure 3. FishBanks server

The server contains following modules:

- Multi Agent System - contains all the agents;
- Parameters .- contains the initialization parameters of the agents;
- User Interface - realizes the communication with the user : setting the parameters and presenting the results;
- Human Interface Listener - realizes the connection to the human players;
- Coming Agents Listener - realizes the migration of the agents;

5.3. Agents of the realization level.

In the previous chapter, the types of agents, which are the elements of the model of the system were presented. At the level of the system realization two others types of agents are introduced:

- **visualiser** - responsible for presenting the results of the operation of the system and recording the history of the simulation.
- **clock** - responsible for synchronization of years and stages.

5.4 Structure of the agent.

The server's process is a multi-thread one where each agent acts as a separate thread. The agent company is the most t complex one as it consists the following elements (Fig. 4.):

- **message receiver** - receives and puts in order messages incoming from other agents;
- **register of current message state** - stores the state of current realised message threads (auction and negotiation);
- **knowledge** - stores data on the state of other agents, history of the game, etc.
- **state** - stores information about the owned resources;
- **decision module** - on the basis of the own state of resources and knowledge and taking into account the negotiated **laws** it selects the actions which are the best for realization of the agent's **goals**, the set of actions to selection is chosen by **strategies module** and the value of the actions is checked by **anticipation module;**
- **message sender** - prepares and send messages to other agents.

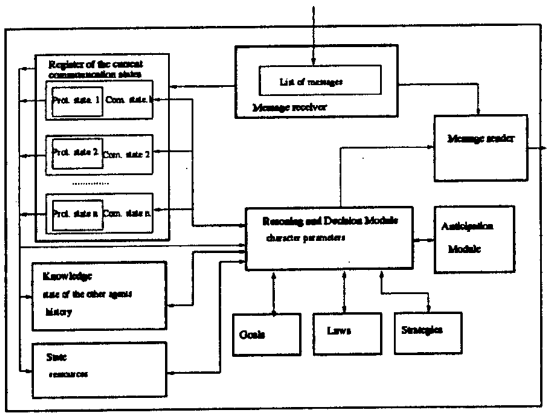


Figure 4. Model of agent

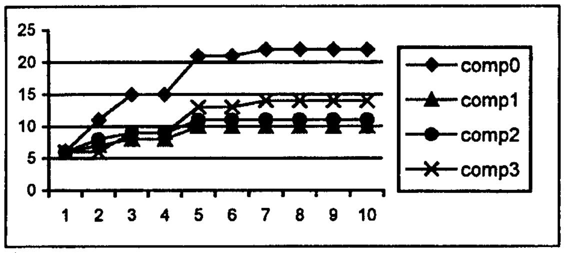
Other agent types are simpler - they do not have a module responsible for remembering the state of communication and their decision-making modules are simplified.

5.5. Remarks

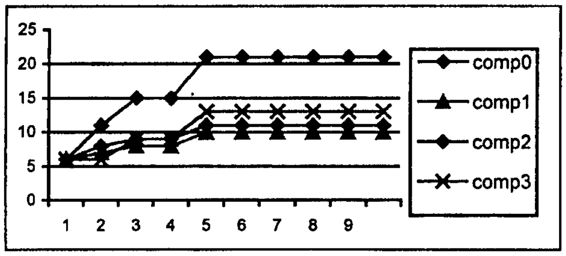
The system Fish Banks is written in Java language(JDK 1.1.6 and library Swing 1.0.3 to make user's interface are used). The sources have about 19000 lines and contain about 140 classes.

6. Results.

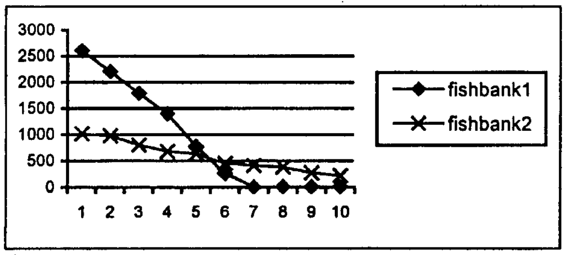
Moreover, we analyzed the change of the state of the system during the simulation and compared the game with negotiations and without it. We did the experiments on the balanced population of the four



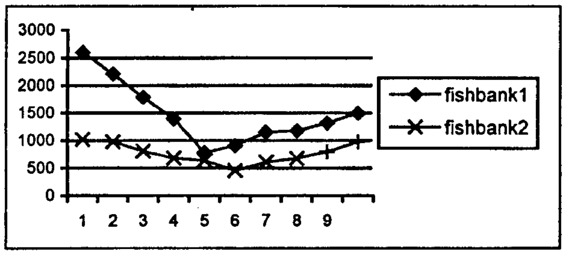
a)



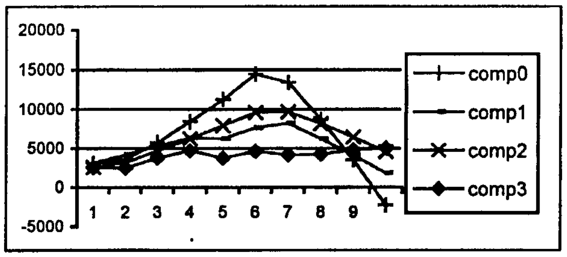
b)



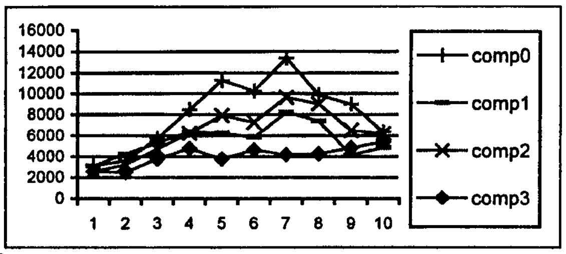
c)



d)



e)



f)

Figure 5. Results.
a,c,e – for the game without the negotiations;
b,d,f – for the game with the negotiations;
a,b - number of ships of companies;
c,d - quantity of fish on fish bank ;
e, f - money of companies.

agents-company (some of them have stronger market preferences, some of them - ecological) and two fish-banks. The agents have following characteristics:

- company0: w = 100, e = 0, k = 0;
- company1: w = 0, e = 100, k= 0;
- company2: w = 0, e = 0, k = 100;
- company3: w = 40, e = 40, k= 20;

We compare the changes in the quantity of the ships and companies' money and the quantity of fish in the fish-banks. The games without the negotiations were finished with the exploitation of both fish-banks. And with the negotiations process the agents made an agreements and limited the level of catching, giving the time for fish to breed.

7. Conclusions

7.1. FishBanks system.

The FishBanks system's goal is to modelize the course of the game so that the behavior of artificial players is similar to the human players' one. But the system indeed may have wider utility. It is possible to change the parameters of the simulation (the prices of ships, fish, interest rate on loans and savings, quantity of the fishbanks and their parameters etc.), which are the constants in the Fish Banks game. We work on the version where the agents migrate between the games (using the Aglets library for migration of code in the computer network).

7.2. FishBanks system and tragedy of commons.

In the FishBanks system we analyze one of the preventing mechanisms of ``the tragedy of commons" - negotiations to agreed mutual commitments. However the system is based on the

FishBank game, still - thanks to the "tragedy of commons" which takes place in very different situations - we may to adopt to the researches on the similar problems.

7.3. FishBanks system and MAS theory.

In the FishBanks system the negotiations guarantee the maintenance of the functional integrity of the system, assuring the sufficient level of the resources. The functional integrity of the system may also be damaged by the lack of the agents with proper capacities. The solution to this problem is to give the agents the possibility to enter to or to leave the system (the agents may be useful in the system or not, and the conditions in the system may satisfy or not satisfy the agents). It is the reason, why we work on the version including the migration of the players between the games.

Bibliography

1. [Antona98] M. Antona, F. Bousquet, C. Le Page, J. Weber, A. Karsenty, P. Guizol "Economic theory and renewable resource management" in Pre-Proceedings of "Multi-agent systems and Agent-Based" Simulation, 4-6 July 1998, Cite de Science - La Villette, Paris, France
2. [Berthet92] Sabine Berthet, Yves Demazeau,Olivier Boissier "Knowing Each Other Better", 11th International Workshop on Distributed Artificial Intelligence, Glen Arbor, February 1992
3. [Bousquet96] Francois Bousquet, Yann Duthoit, Hubert Proton,Christophe Lepage, Jacques Weber, "Tragedy of commons, game theory and spatial simulation of complex systems.", Ecological Economics, Saint Quentin, Mai 1996
4. [Cammarata83], Cammarata S. , McArthur D., Steep R., "Strategies of cooperation in distributed problem solving" in *Proceedings of the 1983 International Joint Conference on Artificial Intelligence*, p875-883, 1993.
5. [Demazeau95] Yves Demazeau "From Interactions to Collective Behavior in Agent-Based Systems", European Conference on Cognitive Science, Saint-Malo 1995,
6. [Durfee87], Durfee, E. H., Lesser V. R., Corkill D.D. "Coherent cooperation among communicating problem solvers", *IEEE Transaction on Computers*, C(11):1275-91.
7. [Georgeff84], Georgeff, M.P. "A theory of action for multi--agent planning" in *Proceedings AAAI-84*, p.121-125, 1984
8. [Hardin68] Garret Hardin "The tragedy of commons", Science, 162:1243-1248, and is accesible in the Internet at http://dieoff.org/page95.htm
9. [Meadows93a] Dennis L. Meadows, Thomas Fiddaman, Diana Shannon, "Fish Banks, LTD. - Game Administrator's Manual", Laboratory for Interactive Learning, Institute for Policy and Social Science Research Hood House, University of New Hampshire, Durham, USA, July 1993
10. [Meadows93b] Dennis L. Meadows, Thomas Fiddaman, Diana Shannon "Fish Banks, LTD.- Materials Manual", Laboratory for Interactive Learning, Institute for Policy and Social Science Research Hood House, University of New Hampshire, Durham, USA, July 1993
11. [Proton97] Proton Hubert, Francois Bousquet, Philippe Reitz "Un outil pour observer l'organisation d'une societe d'agents. Le cas d'une societe d'agents chasseurs agriculteurs." *Actes des 5e Journees Francophones d'Intelligence Artificielle et Systemes Multi-Agents,* La Colle sur Loup, Cote d'Azur, 2-4 april 1997, p.143 - 157
12. [Rouchier98] Juliette Rouchier, Olivier Berreteau, Francois Bousquet "Evolution and Co-Evolution of Individuals and Groups in Environment" in *Proceedings of 3rd International Conference on Multi-Agent Systems* (ICMAS'98), 1998
13. [Sian92], Sati Singh Sian, "Adaptation Based On Cooperative Learning in Multi-Agent Systems" in *Decentralized A.I. vol 3* Eric Werner and Yves Demazeau, 1992 Elsevier Science Publishers B.V.
14. [Turner93] Roy M. Turner "The tragedy of Commons and Distributed AI Systems", in *Proceedings of the 12th Internationall Workshop on Distributed Artificial Intelligence,*

MULTI-AGENT MODEL FOR METRO SCHEDULING

Stanislav V. Mikoni

St.Petersburg State Transport University
E-mail: mikoni@pgups.spb.ru

Abstract

Problem of automatic metro scheduling is considered. The task is very complicated. High dimension of the task does not permit to solve it by classical methods. It is offered to solve the task with the application of Multi-Agent System technology. Trains running in metro line are considered as a controlled environment. Metro stations controlling train movement are presented as agents of bottom level. The functions of the agent are determined on the basis of the role of the station in train movement control. Bottom level agents use knowledge about the line and its divisions from agents of high levels. The offered technology allows finding acceptable variants of the metro schedule. The prototype of the system was successfully tested for one of the lines of St.Petersburg metro.

Keywords: *metro line, terminal, turnaround station, leave/return station, night parking station, train, train situation, line schedule, circled schedule, Multi-Agent system.*

1. Introduction

The task of scheduling is of high complexity. It is determined by the variable numbering whose meanings should be found while scheduling. Each variable corresponds with the departure time of *i-th* train from *j-th* station. The number of these variables is proportional to the number of stations n, the number of trains N running on the line and the number of circles D/T performed by each train for a working day: $2*n*N*(D/T)$.

Here D – time of metro operation, T - circle time of a train running on the whole line. Coefficient 2 takes into account two directions of train movement on the metro line.

According to the formula the number of variables for a real metro line is estimated in thousands ones.

The task of metro schedule making up would have been simple solution if the interval between all neighboring trains had remained constant within a working day. But it changes under the influence of the following factors:

- change of intensity of a train movement dependent on the passenger flow fluctuations during the time of metro operation;
- train leave from the line for maintenance service and its return to the line;
- reordering of trains according to their numbering performed at the working day end for their location at the appointed night parking places.

Thus the task of metro scheduling is an optimizing one. The interval uniformity of train movement is taken as the criterion. It is the function of time. The above factors, which have numerical value, are used as restrictions. According to high complexity of the task it is impossible to apply mathematical programming models for its solution. Heuristic approaches do not give desirable results either [1].

In this paper the solution of the formulated problem is offered with Multi-Agent technology application. Multi-Agent model for metro line movement allows not only to execute decomposition of the task on n of subtasks, but also to ensure their activity by corresponding them with agent's [2].

We first consider the subject field model construction in MAS terms [3]. We then give the theoretical basis for the agent function synthesis. Finally we describe two fragments of the Metro Scheduling System (MSS) and discuss MAS properties of the proposed model.

2. Metro Line as Multi-Agent Model

There are two items participated in the metro line movement: metro stations and trains running on the line. Within object-oriented programming sense it may be possible to consider them as objects. Concerning control aspect these objects are distinguished as active and passive ones respectively. From MAS point of view let us assume metro stations as agents of the system. The trains form the environment of the system.

To construct Multi-Agent Model for Metro Line let us consider some basic properties of a train and a station.

The following features characterize each train:

- route number;
- order number among the trains running on the line for night parking organization;
- stations of today and tomorrow night parking;
- the station and the approximate time of line leave for maintenance service and return from it;
- approximate departure time from today night parking station;
- approximate arriving time to the next night parking station.

The basic MOTION function of the train as a movement participant is not directly used, as it does not take part in movement organization.

Concerning their role in train movement control metro stations are distinguished as linear, terminal, turnaround and leave/return ones (Fig.1). The types of stations differ by their resources, which are understood as train parking places and track layout. Linear stations, as most simple, have only to delay train departure. The terminal has in addition to the above function to control the order of train departure to the opposite movement track. The turnaround station can also reorder trains, but it makes this by means of inserting of the arrived train into a train pair running in the opposite direction. The leave/return station in addition to the above functions can carry out the function of taking the train off the line and returning it to the line.

Thus the above-described stations make decisions for movement control inherent to agents in the MAS [4]. According to Demazeau [4] the basic kind of interactions between metro MAS agents is implemented by communication through trains (the environment), using them as a shared resource.

However, metro stations have no full information about the line. Knowledge about the line and its divisions used by the station agents for decision-making is distributed between the line agent and division agents. The line agent besides provides initialization of representatives of the TRAIN object and controls the event processing sequence in the MSS. Thus, the Agent metro model has the hierarchical structure. Metro stations form the bottom control level of the MSS. The intermediate level of the model consists of division agents. The top model level consists of the single line agent. Agents of high levels give division and line information for agents of the bottom level. Besides the line agent is the monitor of MSS.

3. Agent Structure

As the object of the program system each agent of the MSS consists of a database and a set of functions. The database of the station agent contains the following static information:

- station name;
- station type (terminal, turnaround station, leave/return station);
- left and right neighboring stations;
- left and right adjacent divisions (division length, average time of train running on the division);
- station resources (track layout and the number of parking places);
- parking place type (even or odd day, constant or reserve).

The parking place state (vacant or engaged) relates to the dynamic information type.

The functions of the station agent are determined by its role in the train movement control. Each station has a DEPARTURE function. It calculates the train departure time from the station on the basis of global and local average intervals of train movement in a current period of time.

The turnaround station besides executes 2 functions: ENTER and EXIT. First of them makes the decision concerning the direction of train movement, i.e. to continue the movement or to change it into the opposite direction. In the second case the train enters the station dead end. The EXIT function determines the moment of the train transfer from the dead end to the opposite direction track in order to insert the train to the proper place of the running train sequence.

The terminal executes the CIRCLE function determining the sequence of train departure in the opposite direction.

The leave/return station executes the LEAVE and RETURN functions. The first of them determines the moment when the train leaves the line for maintenance and the second one determines the moment when the train returns to the line after its maintenance.

The input data for function execution are the train situation involving trains being at the station and its adjacent divisions. The result of processing the current train situation by the station agent is the record of pairs: «train number, departure time». It forms output data array. The set of these arrays represents the metro line schedule allocated at all stations.

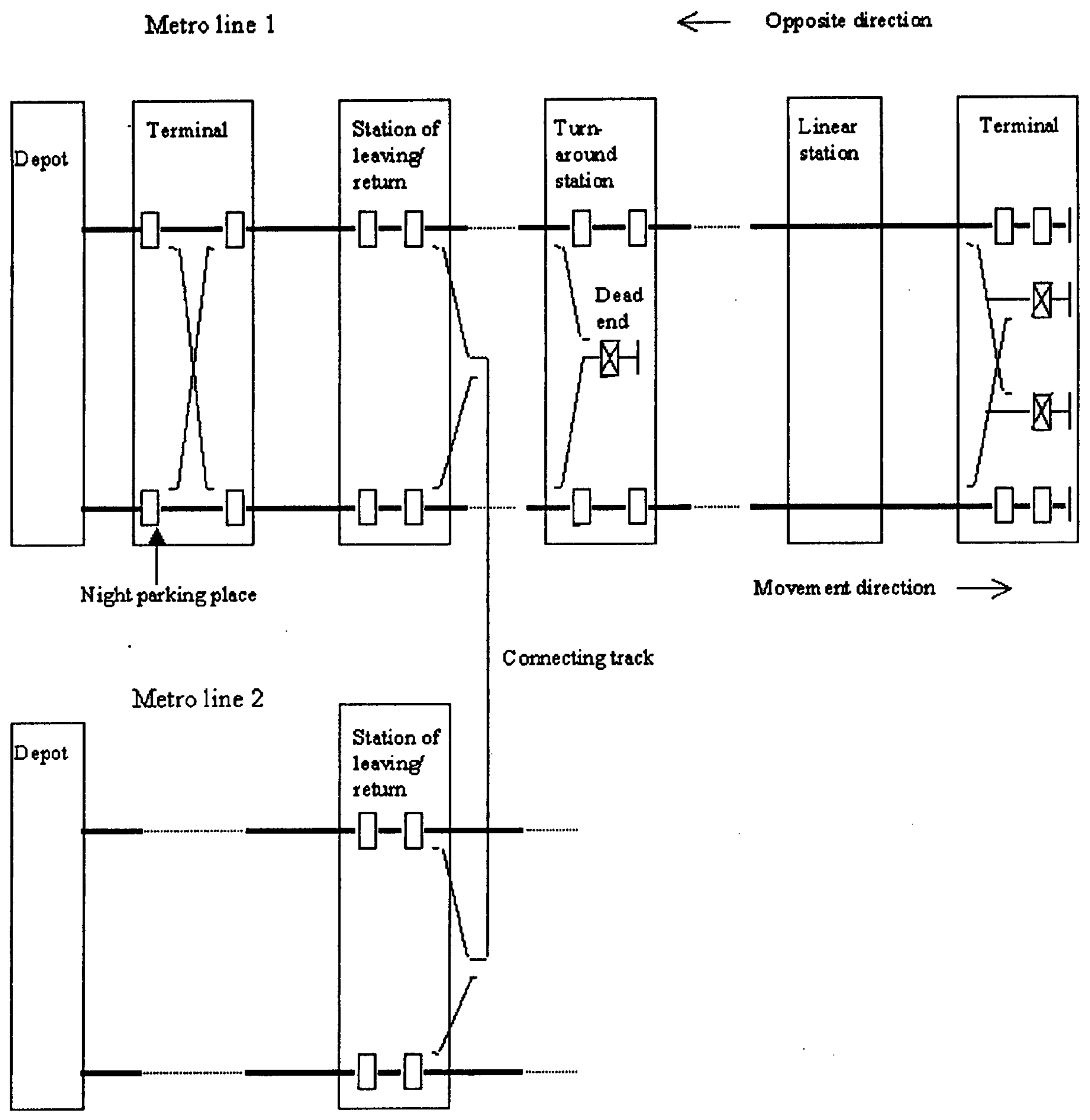


Fig. 1 Metro line model

The database of the division agent contains the following static information:

- division name;
- set of stations between two neighboring night parking stations;
- division length;
- average time of train running on the division.

The dynamic information is the set of trains running on the division at the current time.

The division agent executes one INTERVAL function calculating local average interval of train movement on the line division.

The database of the line agent contains the following static information:

- line name;
- set of stations belonging to the metro line;
- line length;
- circle time of trains running on the line;
- planned time of the movement beginning;
- planned time of the movement end;
- minimal interval between neighboring trains;
- time of the night parking beginning;
- maximal diversion relatively the line leave time by the train.

The dynamic information is set of trains running on the line at the current time.

In addition to the INTERVAL function calculating the global average interval of train movement the line agent executes following functions: INITIALIZATION, NUMBERING, START, CONTROL, ORDER. The INITIALIZATION function sets the planned tasks to all representatives of the TRAIN object. The NUMBERING function establishes relative numbers of trains remaining on the line for locating them at night parking places. The START function forms initial queue of TRAIN representatives. The CONTROL function exercises the sequence of processing of TRAIN representatives. The ORDER function estimates the end of lexicographic reordering of trains before their location at night parking places at the working day end.

4. Metro Scheduling

The metro line schedule is calculated by means of Multi-Agent Model behavior simulation.

There are 3 stages of train movement during a working day:

- beginning of movement on the line;
- interval management;
- location of trains at night parking places.

The initial data for modeling are contained in the circle schedule. They represent a set of the planned tasks for each train:
- night parking place and the departure time from it;
- place of the future night parking and the time of arrival to it;
- technical maintenance place;
- station and the time of line leave for technical maintenance;
- station and the time of the return to the line after technical maintenance.

The line agent inserts these data to all representatives of the TRAIN object. Then it carries out the relative numbering of trains concerning the future night parking places.

The line agent controls the functioning of the system. The CONTROL function analyses the first element of the train queue and activates the agent representing the station of departure for the chosen train. The result of processing of the TRAIN object is time of its departure fixed in the output array of the station agent. According to the value of the departure time the CONTROL function finds a new place in the queue for the TRAIN object on the basis of lexicographic ordering.

Let us consider the metro line Multi-Agent Model functioning at each stage of train movement during metro operating time.

4.1. Start stage

At that stage the station agents execute 2 goals: maintenance of the train departure interval uniformity and alignment of the first trains starting time from all night parking stations.

The start stage is carried out by the system in 2 steps. At the first of them the line agent forms the initial train queue by START function. At this step the planned train departure time from night parking place is calculated. On the 2-nd step the processing of queue elements is carried out with fixing of a real departure time in output data array of the station agent. At this stage the real departure time coincides with the planned one.

Each element of the TRAIN object queue consists of three elements: «a planned departure time, train number, departure station». The queue is ordered according to the increase of the departure time of trains. It consists of 2 parts. The trains participating in the start from night parking places up to 6 a.m. make up the first part. The trains started after 6 a.m. from night parking places or depot form the second part of the queue.

The planned departure time of trains forming the second part Q_2 of the queue Q is directly determined on the basis of the circle schedule of the line. The planned departure time of trains forming the first part Q_1 of the queue Q additionally takes into account the following factors:
- beginning t_0 of movement on the line;
- distribution of night parking places on the basis of sequence of train exit from the station;
- average train departure interval from night parking places.

The division agent on the basis of the division length and the number of trains calculates the last parameter concerning night parking places.

The beginning of movement of the 1-st train from each night parking place should be close to the time of the beginning of movement on the line. The delay of the 1-st train departure is caused by the occupation of the next night parking station and is to be minimized.

4.2. Interval management

At this stage the station agents also execute 2 goals: maintenance of uniformity of train movement intervals and maintenance of time for line leaves by trains for technical service in the range of ±15 minutes concerning planned leave time.

At the arrival of the train to the line the interval between neighboring *i-1* and *i* trains decreases two-fold:

$$T_{int,i-1,i}=(t_{dep,i-1,l}-t_{dep,i,l})/2, \qquad (1)$$

and for the train leave from the line it increases two-fold:

$$T_{int,i-1,i}=2(t_{dep,i-1,l}-t_{dep,i,l}). \qquad (2)$$

Here $t_{dep,i,l}$ - departure time *i*-th train from *l*-th station.

The average interval $T_{int,line}$ between neighboring *i-1* and *i* trains at the arrival of an extra train to the line is recalculated according to the formula:

$$T_{int,line}:=T_{int,line}\, N_{line}/(N_{line}+1), \qquad (3)$$

and the time of leave of the line - according to the formula:

$$T_{int,line}:=T_{int,line} N_{line}/(N_{line}-1), \qquad (4)$$

The recalculation of the interval is carried out at each arrival/leave event.

As the interval of train movement on the line $T_{int,line}$ is average in relation to intervals $T_{int,i-1,i}$, $i=1,\dots,N_{line}$, the alignment is either reduced to the interval $T_{int,i-1,i}$ at $T_{int,i-1,i}>T_{int,line}$, or is increased at $T_{int,i-1,i}<T_{int,line}$.

The reduction of the interval $T_{int,i-1,i}$ is obtained by the delay of the previous train leaving *j* station and the increase - by the delay of the subsequent train coming to *j-th* station.

As the previous train is already started from the station the delay is performed by the correction of its departure time in output data array of the station. The delay of the coming train is performed by the recount planned departure time to the real one according to the formula:

$$dT_{int}=\left| T_{int,i-1,i}-T_{int,line} \right| \qquad (5)$$

The alignment of intervals of train movement can be instant or smooth. The instant alignment of the interval is carried out during 1 step by the delay of the previous or the subsequent train on dT_{int}. If this difference is great, it is possible to distribute the delay between N_{div} trains occupying this division:

$$dT_{int}=\left| T_{int,i-1,i}-T_{int,line} \right| /N_j. \qquad (6)$$

The second goal of this stage is achieved by forecasting the conflict for connecting track resource at the leave/return station. The terminal agent compares planned time of leave for the analyzed train with the time of leave (return) of other trains. If the time of line leave for two trains coincides, the delay of the analyzed train at the terminal is carried out.

4.3. Location of trains at night parking places

As the result of train leaves from the line and returns to it the order of their following is broken. So at this stage reordering is necessary. Its goal is to send each train to destined night parking place. This process is formulated as the task of lexicographical ordering of trains under their numbers nominated concerning of their future night parking places.

The reordering of trains is carried out by the agents representing terminals, turnaround stations and, as an exception, leave/return station. The agent of a turnaround station (or a line leaving station) chooses the direction of further movement (direct or opposite) of the analyzed train. The agent of terminal chooses the sequence of train departure.

The decision making for future train movement is carried out on the basis of the calculation and comparison of cyclic differences of train numbers for current train situation at this station. The train situation is supposed as a set of trains, which are at the station and on divisions of the line adjacent to the station.

The cyclic difference is calculated for those pairs of trains which may be formed according to the relation «previous, subsequent» (N_{pr},N_{ss}). There are 2 groups of pairs: preserving and changing movement order. Terminals and turnaround stations form these groups.

As the module of a cyclic difference greatest number of relative numbering of trains is accepted. The cyclic difference $dN=(N_{pr}-N_{ss}) mod N$ is calculated according to the following rule:

if $N_{pr}\leq N_{ss}$, then $dN:=N_{ss}-N_{pr}$,
else $dN:=(N-N_{pr})+N_{ss}$.

The variant of movement order chooses whose characterizing group has the pair with minimal cyclic difference.

After every action of train reorder the line agent checks the lexicographical order of trains on the line. If the order is achieved the process of train reorder is over and trains are set for night parking places.

Let us consider the decision making by turnaround station and terminal agents. It is executed for further train movement variant choice by the cyclic difference analyses using.

4.3.1. Decision making by turnaround station agent

The turnaround station agent makes decision on the basis of train situation at the turnaround station after train arrival. The model of that station used for the cyclic difference calculation is shown in figure 2.

The train situation at the station consists of 4 trains and the train array:

Nf - train number for the train arrived to the station;

Nif - train number for the train following the arrived train;

Nof - train number for the train sent from the station;

Nob - train number for the train sent from the station to the opposite direction;

Nib[*i*] - the train number array for trains running on the division previous to the station on the opposite track.

Cyclic differences *dN* showing the existing movement order:

$dof = Nof\text{-}Nf$, $dif = Nf\text{-}Nif$, $db = Nob\text{-}Nib$..

Cyclic differences *dN* showing the changing of movement order:

$dob = Nob\text{-}Nf$, $dib = Nf\text{-}Nib$, $df = Nof\text{-}Nif$.

There are some variants of the movement order changing. They differ by the train pairs running in opposite directions into which train *Nf* must be inserted. To find the best variant of insertion in respect to cyclic difference value all possible pairs formed on the basis of the array *Nib*[*i*] are considered. The first of them is the pair (*Nob*,*Nib*[1]) and the others are formed as (*Nib*[*i*],*Nib*[*i+1*]), $i=1,...N_j$-1. Here N_j - the number of trains running on the *j-th* division adjacent to the station. If the movement order changing decision is taken the train *Nf* is sent to the dead end. It should wait there while the chosen pair of trains will arrive to the turnaround station.

After the calculation all cyclic differences *dN* the minimal one is chosen:

1) $dN_{s,\ min}$=min{*dof, dif, db*} - for the decision of movement order preservation;

2) $dN_{c,\ min}$=min{*dob, dib, df*} - for the decision of movement order changing.

The turnaround station agent makes the decision according to min{$dN_{s,min}$, $dN_{c,min}$}.

4.3.2. Decision making by terminal station agent

The terminal agent chooses the train, which should be departed after the current train arrival, by using the train situation analysis. The train situation on the terminal consist of the following trains (Fig.3 and Fig.4):

Nf - the train number for the train arrived to the station;

Nob - the train number for the train departed from the terminal into the opposite direction;

NT1 - the train number for the train occupying parking place 1 in the dead end;

NT2 - the train number for the train occupying parking place 2 in the dead end;

Nif[*i*] - the train number array for trains running to the terminal from the adjacent division.

The condition of changing the order of movement from the terminal is the existence of 2 or more parking places. There are 2 variants of these allocations:

1) 1 dead end having 2 parking places (Fig.3);

2) 2 dead-ends, each having 1 parking place (Fig.4).

The first variant is characterized by the consecutive parking place order. It may be described as a stack (the last to come, the first to go).

For this variant there are the following possible train situations at the terminal:

1.1. Trains occupy both places. There are no alternatives for the choice. The departure of the train *NT1* is carried out.

The second variant of allocation is characterized by the parking place parallelism. For this variant there is possible following train situations at the terminal:

2.1. Trains occupy both places. There are 2 possible alternatives for the choice:

2.1.1. The departure of the train *NT1* is performed.

2.1.2. The departure of the train *NT2* is performed.

2.2. One parking place is vacant. There are 2 alternatives for the choice:

2.2.1. The departure of the train *Nf* is performed.

2.2.2. The departure of the train *NT1* is performed.

2.3. Both places are vacant. There are 2 alternatives for its choice:

2.3.1. The departure of the train *Nf* is carried out. The movement order is preserved.

2.3.2. The train *Nf* is sent to a parking place for the movement order changing.

Thus decision making for these train situations differs only when both places are occupied by trains. Obviously the train situation 1.1 may be considered as a particular case of the train situation 2.1. Due to this it is possible to use

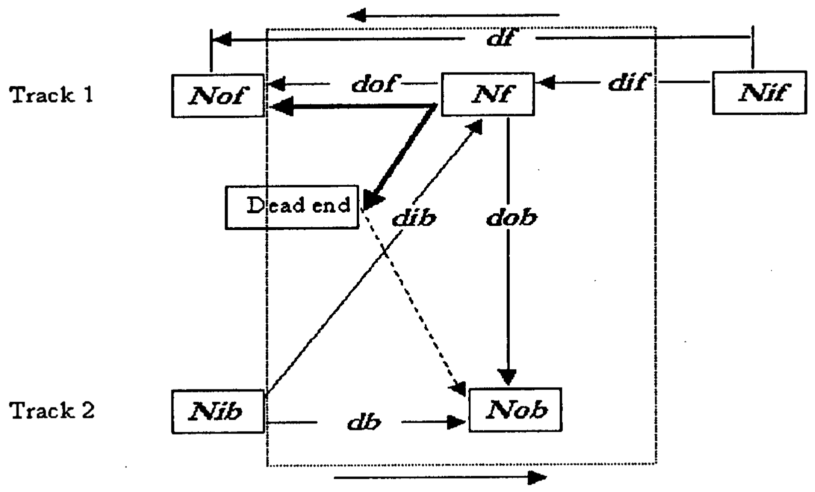


Fig.2. The train sitution for the turnaround station

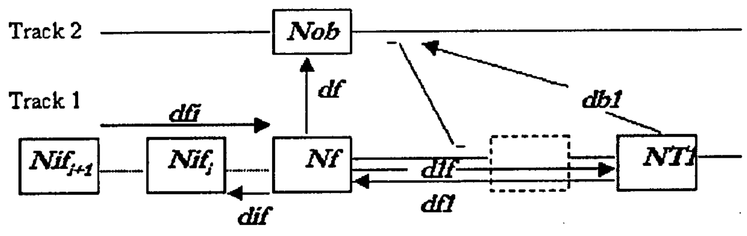


Fig.3. The train sitution for the terminal with 1 dead end

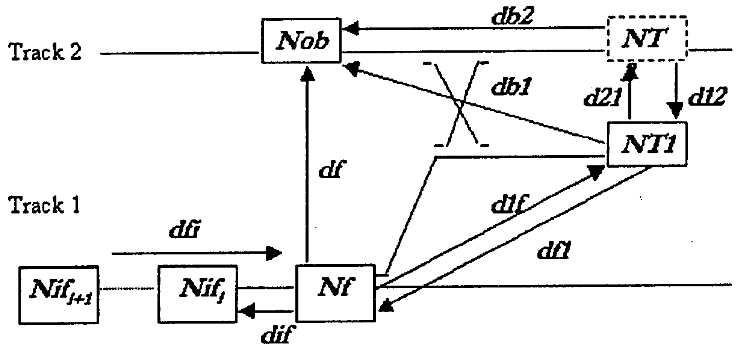


Fig.4. The train sitution for the terminal with 2 dead ends

common algorithm of decision making for both variants.

The algorithm uses the following formulae:

1. There are 2 dead ends. Both places are occupied:

$db1=Nob\text{-}NT1;\ d12=NT1\text{-}NT2 \rightarrow$
$dN_{12}=\min\{db1,d12\}$;
$db2=Nob\text{-}NT2;\ d21=NT2\text{-}NT1 \rightarrow$
$dN_{21}=\min\{db2,d21\}$;

If $dN_{12}<dN_{21}$, then the train $NT1$ is departed from the station, else it is allocated to $NT2$.

2. Both places are vacant:

$df=Nob\text{-}Nf;\ dif1=Nf\text{-}Nif[1] \quad \rightarrow$
$dN_{bf}=\min\{df,dif1\}$;
$dif=Nif[i]\text{-}Nf;\ dfi=Nf\text{-}Nif[i+1] \rightarrow$
$dN_{if}=\min\{dif,dfi\}$, $i=1,\ldots,nb$.

Here nb - the number of trains following the train Nf.

If $dN_{bf}<\min dN_{if}$, then the train Nf is departed from the station, else it is sent to the dead end for waiting for train pair $(Nif[i],Nif[i+1])$ into which it must be inserted.

3. One parking place is vacant:

$df=Nob\text{-}Nf; \quad df1=Nf\text{-}NT1 \quad \rightarrow$
$dN_{Nf}=\min\{df,df1\}$;
$db1=Nob\text{-}NT;\ d1f=NT1\text{-}Nf \quad \rightarrow$
$dN_{NT1}=\min\{db1,d1f\}$;
$dif=Nif[i]\text{-}Nf;\ dfi=Nf\text{-}Nif[i+1]\rightarrow$
$dN_{if}=\min\{dif,dfi\}$, $i=1,\ldots,nb$,

If $\min\{dN_{Nf}, dN_{NT1}, dN_{if}\}=dN_{Nf}$ then the train Nf is departed from the station.

If $\min\{dN_{Nf}, dN_{NT1}, dN_{if}\}= dN_{NT1}$ then the train $NT1$ is departed from the station.

If $\min\{dN_{Nf}, dN_{NT1}, dN_{if}\}= dN_{if}$, then the train Nf is sent to the dead end.

5. Experiment results

On the basis of the offered metro line Multi-Agent model two fragments of the Metro Scheduling System (MSS) had been designed.

The objective of the first fragment was to reorder train sequence from the arbitrary order to lexicographical one. The main problem of that task is to achieve a coherent behavior of agents. It was discovered the task solution depends on a set of the following factors:

- time of solution;
- jointed action of both agent type – terminal and turnaround station agents;
- proper distribution of turnaround station agents on the line;
- train delay on both tracks before the turnaround station in order to insert the transferring train into the proper train pair running on opposite track;

As the reorder task has no single solution it is multifold solved by variation of above factors. The coherent behavior of agents is achieved when the sufficient combination of the factors takes place. It had been experimentally received of 7 variants of solution from 176 factor combinations for the line example.

No single task solution permits to choose the best reordering result among received different variants. The last ones for the example have been estimated relatively to factor values.

The objective of the second MSS fragment was to simulate movement beginning and interval alignment during train starts and its line leaves/ returns. The program automatically makes up and displays the line schedule during train movement simulation.

The MSS has some restrictions:

- the metro line model can consist of up to 25 stations;
- maximum number of trains on the line may be up to 60;
- circle time of a train on the line may be up to 150 min.

Before simulation following system installation is performed. Initially the system is set for the concrete metro line. After that trains are set to night parking places. The opportunity of line model updating is provided.

The system prototype was successfully tested for 2 variants of the 4-th line of St.Petersburg metro. They consist of 10 and 12 stations at the line respectively. The line schedule without the last stage was built during some minutes. The schedule quality is acceptable and can be improved by change of system variable meanings.

The second MSS fragment is created at C++ language.

6. Discussion

It had been initially applied a natural approach to solve the complicated task, i.e. the decentralized model have substituted for the centralized one. It had been decided to solve the task according to "from down to up" principle. For that reason it is possible to relate the decentralized model to Distributed Artificial Intelligence (DAI) field. Hence it is no sufficiently to relate that model to MAS field. An agent of MAS must possess such features as autonomous, interactive and proactive ones.

At object-oriented programming point of view let us consider the first feature. The event in MSS is thought as arrival of the train to the station. The station agent makes decision by event processing. Each event-processing agent depends on no rest agents. The line agent controls no trains. Its role is only to form train queue in order to organize station agent activity. In fact each agent is autonomous one. Its decision is determined by only own train situation.

At the first look there are no visible relationship between agent of the system. Hence there are some occasions when relations are obvious. They exist between station and division agents. Division agent participates in solving of 2 tasks. It calculates the planned departure time of trains starting at movement beginning in order to align intervals between trains. Another alignment task is solved when trains leave/return the line. There are 2 ways to solve the problem. If exact time of trains leaving/returning the line is known it is used for other trains departure time calculation. If exact time of train leaving/ returning the line is unknown when leave/return event is occurred the division agent recalculates early-calculated departure time of trains.

As against vertical agent interaction horizontal one actually is very weaken. It is realized indirectly in controlled environment (trains running on the line). In spite of the fact that horizontal interaction is weak, one agent cannot solve tasks of the system. For example the train reorder task demands not only participation community of agents, but also their heterogeneity.

The problem of agent interaction in the reorder task can be solving by two ways: external and internal ones. In the MSS the external way is chosen. It is realized through allocation to the agents of spheres of their actions. This choice is adequate to a task of planning admitting consideration of several schedule variants received by model factor variation. For operative solving of the task the internal way of agent interaction realized through mutual agent analysis is more preferable. Hence that way is more expensive.

The task of scheduling does not require evolution of agents. Therefore station agents have no feature of proactivity. Here Ockham principle is pertinent: "Entities should not be multiplied unnecessaraly".

At artificial intelligence point it may assume a function calculation as agent reasoning. As agents of MSS carry out the simple enough protocols ones have rather the reactive type. According to Braspenning [5] it may consider agents of MSS as plant-like ones. Its interaction may be thought as external coordination. However all classifications are relative.

7. Conclusion

In this paper we have advanced an approach for automatic metro line scheduling. It is proposed MAS model for complicated task solving [6]. Two fragments of the Metro Scheduling System (MSS) had been designed.

The following basic results are received.

1. The subject domain is structured on agents and environment. As agents metro line stations are accepted and environment is thought as train running on the line. First are in a static state and second are in dynamic one. Thus the MSS model feature compared the most MAS systems [7] is dynamic environment and static agents.

2. It is developed the architecture of agents. They solve several tasks of metro movement control. On the basis of models of interval alignment and train reorder the appropriate functions of agents are designed.

3. It is experimentally shown that tasks of the system are solved only under condition certain combination of agents. The structure specifies diversity of agent roles in order to rich common goal of the community. Such community of agents gets emergence property.

4. The coordination of agent actions plays an important role in the task solution. At the interval problem solution the coordination is achieved with application of the agents-coordinators. The latter provide station agents with average interval of train movement information. The lexicographical ordering problem is solved by allocation of station agent action spheres.

5. It is necessary to note that metro schedule made up according to the offered technology is not optimal. It is only the result of the adequacy to given restrictions. On that reason mechanisms is used for providing to return to early stages of decision taking. Thus limited set of variants of metro schedule may be generated and the best variant may be selected. Such solution of the problem is quite allowable for the task of planning.

We should like to denote the MSS might be not only for the scheduling problem solution but also to design track layout of the metro line stations for movement optimization goal.

The similar MAS model can be applied for solving of tasks in another subject domains, for example in communication field (packet switching, etc.).

Acknowledgments

I would like to thank all the people with whom I had worked and without whom such a paper could have been written: Ilya Petrov and Vagif Mamedov.

References

1. S.V.Mikoni. Decentralized Approach to Transport System Scheduling. The theses of International Conference «Regional Informatic-98». June 1998. St.Petersburg. Russia. pp. 92-93 (in Russian).
2. S.V.Mikoni, I.V.Petrov. Object-Oriented Model for Metro Line. The theses of International Conference «Regional Informatic-98». June 1998. St.Petersburg. Russia. pp. 92-93 (in Russian).
3. V.I.Gorodetski, M.S.Grushinski, A.V. Khabalov. Multi-Agent systems (overview). News of Artificial Intelligence. Moscow. A.I.Association N2, 1997 (in Russian).
4. C.Baeijs, Y.Demazeau. From spatial databases to organised Multi-Agent Systems. In Proceeding the International Workshop «Distributed Artificial Intelligence and Multi-Agent Systems»/ 15th-18th June 1997. St.Petersburg. Russia. pp. 182-191.
5. P.J. Braspenning. Plant-like, animal-like and humanoid agents and corresponding Multi-Agent Systems. In Proceeding the International Workshop «Distributed Artificial Intelligence and Multi-Agent Systems»/ 15th-18th June 1997. St.Petersburg. Russia. pp. 64-77.
6. P.Pesin, C.Tahon, V. Tarassov. Multi-Agent Architecture for the Activity Control of Complex Industrial Systems. In Proceeding the International Workshop «Distributed Artificial Intelligence and Multi-Agent Systems»/ 15th-18th June 1997. St.Petersburg. Russia. pp. 182-191.
7. A. Adamatzky, O.Holland. Swarm intelligence: representations and algorithms. In Proceeding the International Workshop «Distributed Artificial Intelligence and Multi-Agent Systems»/ 15th-18th June 1997. St.Petersburg. Russia. pp. 13-22.

A MULTI-AGENT SYSTEM FOR SCIENTIFIC PROBLEM SOLVING

V. Negru, C. Sandru and M. Rotaru

Department of Computer Science, West University of Timisoara,
Bd. V Parvan, 4, 1900 Timisoara
email: {vnegru, sandru, mrotaru}@info.uvt.ro

Abstract

The goal of this paper is to propose a multi-agent architecture intended to be used for problem solving (especially complex scientific problems). Our study refers to a model of cooperating agents involving expert agents which use a task oriented reasoning. We developed a prototype where the communication is solved using the KAPI environment and KQML communication language. The reasoning model for agents was implemented in the CLIPS expert system shell.

Keywords: *distributed problem solving, multi-agent systems, rule and object knowledge representation, communication languages.*

1. Introduction

Generally, complex problems solving combine programs written in different languages and running concurrently in different computers [16,17] and are characterized by: difficulties tasks that need different problem solving skills; concurrent and continuously evolving models; concurrent and distributed activities with high computation in communication rates; multiple expertise; imprecise, incomplete and dynamical knowledge; multiple conflicting goals; large data bases processing; real-time constraints etc.

A possible range of application of distributed problem solving techniques are the scientific problems which can be decomposed in sub-problems (independently or not) which is possible to be solved separately. Our goal is to develop a problem solving system based on a multi-agent architecture to be used when solving scientific problems.

Related work already exists to develop such systems and we can cite here general-purpose architectures such MIX [9]. This system provides a network model and an agent model to structure the solving process. ARCHON [2] is another example of a general-purpose system intended to supervise industrial applications. Many ideas from this system represent basics on MAS architectures. Some other systems are more specific and involve artificial intelligence notions like expert systems. We mention here SYNERGIC [18] which uses *the acquaintance* model to represent the interactions between expert agents, together with communication facts in order to exchange knowledge. DESIRE [1] framework uses the task model to specify the multi-agent system competencies.

NESS (Non-linear equation systems solver) [13] is an intelligent front-end for non-linear equation systems solving, developed in CLIPS. Starting from the features of the system and of the numerical methods, human expert use domain knowledge (numerical analysis) and heuristics to:

- choose the most suitable method
- interpret the results (intermediary and final)
- restart the solving process in the case of failure.

NESS uses a task-oriented reasoning. For the domain of solving differential equations systems there is the ODEXPERT system [10] having the goal to provide expertise when solving such problems. Also, the EPODE [6] solving environment, for differential equations and systems of equations, offers helpful features regarding user interface and expertise.

Our work is centered on developing a skeleton of a distributed multi-agent system, which can be extended, with a new collection of communicative specialized agents working on a specific goal. We implements a task concept [3] which we feel useful when solving scientific problems but we also work

on some agent communication issues and how they could be efficiently implemented.
We developed a multi-agent prototype based on NESS extension.
A general problem solving system, providing a remote interface, is NetSolve [14]. But this system is mostly concerned about how to provide a service and how to specify any general request apart from our intention of having a powerful non-linear problem solver. NESS is rather intended to provide services for NetSolve engine.

2. A description of the NESS problem

The formal description of the problem is done by:
$F(x) = 0$ where $F : D \subset R^n \to R^n$
We consider the problem of numerically solving the nonlinear equation system, determinating
$x \in D$ such as $F(x^*) = 0$
There are various methods to do that ranging from more general to more particular methods. No method can solve any system and there are systems, which cannot be solved using known methods.
From the numerical solving methods, we have selected some classical ones with high performances (Classic Newton, Discretizated Newton, Secant, Steffensen, Parallel Lines, etc.) or others relative recently developed (Broyden, Friedman, Conjugated Gradient with Reeves Balanced Factor, Successive Over-relaxation, Alternating Directions Iteration}, etc.), as well as special methods, generally useful for solving equations with differences resulted from continuous problems discretization (for example, the equations with differences corresponding to the weakly nonlinear two point boundary problem).
The methods for solving nonlinear systems of equations have some certain intrinsic characteristics (which do not generally depend on the particular system, which has to be solved); such characteristics are crucial elements of the general solving strategy. There have also been developed special methods for certain classes of problems, using some specific particularities of the problems and to get enhanced performances by these means.
The principal characteristics are: the convergence rate [5], the computing effort per one iteration [4], the possibilities to exploit the sparsity and the numerical stability.
The decision about which method is best fitted for solving a given system must be made considering these characteristics of the methods. It is obvious that -for instance- for a sparse system should be tried a method that can exploit the sparsity of the Jacobian. Some other times, the problem is more difficult; for instance, for a badly conditioned system it is not recommended a method which needs the inversion of the Jacobian, but even finding out that a system is badly conditioned is a problem not so easily to be solved.
Beside these characteristics of general order (convergence speed, computing effort, possibilities to exploit the sparsity) there are many methods with special characteristics, appropriate for particular nonlinear systems. This is the case of the nonlinear systems proceeded from the discretization of nonlinear differential systems. For example, the problem of heat conduction with nonlinear boundary conditions, the weakly nonlinear two point boundary value problem, and so on.
Solving nonlinear problems also involve a series of collateral extremely important problems; for example, choosing the initial iteration [15], error evaluation [7], possibility of parallelization, the problem of finding all solutions in given a rectangular domain, etc.
The methods considered in our work are iterative methods with one step (only one method which make exception is the secant method, a method with two steps), having the form:
$x_{k+1} = \Phi(x_k)$, x_0 - given, Φ - function of iteration.
Generally, there is not a unique recipe in solving the non-linear equation systems (non-linear simultaneous equations) [12].
To choose a particular method one should take into account several external aspects: the form of the system, knowledge's about a "good" iteration, structural characteristics (the quality of the Jacobian, condition number), etc. Other elements, that influence the solving process, are the precision, the memory space, and the convergence speed [13].

3. Intelligent front-end

The overall architecture of the NESS problem solving system is presented in the figure 1.
The arrows indicate relations between components, which will be detailed below.

The entire system is build up on the axe User Interface - Expert – Solver where the Expert is a central decision point.
User Interface is the place where the user interacts with the rest of the modules. The user may:

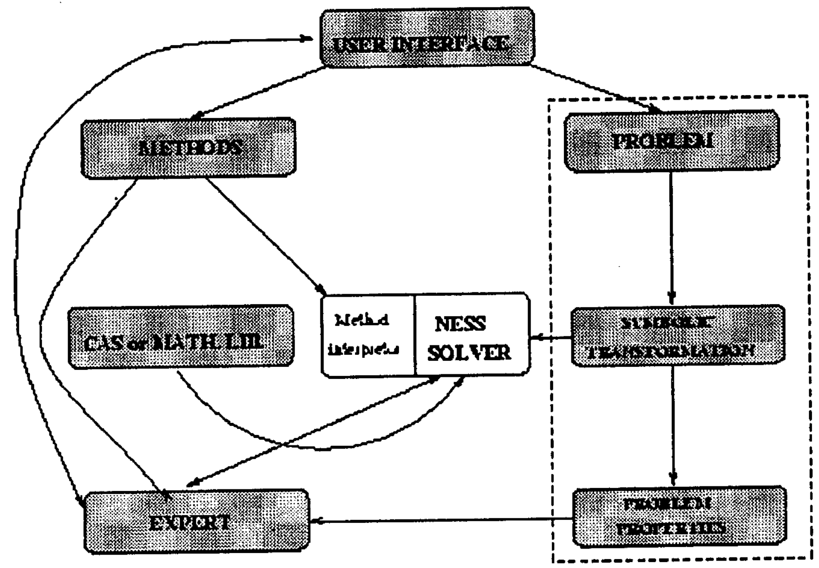


Figure 1: NESS' modules

1. Edit a system of equations
2. Edit a method
3. Specify run options like: desired method to solve the problem, maximum number of iterations, a particular CAS to use, etc
4. Specify preferences concerning the degree he wants to be signaled on the evolution of the solving process.
5. Query the NESS Expert about solving suggestions.
6. Start/Stop the solving process.
7. Query the NESS Expert about its reasoning process.
8. Ask to have a graphical display of the solving iterations.

As indicated in the figure, the Interface have a bi-directional communication link with the Expert, in order for the above to work. The Expert is able to retrieve solving aspects from the Solver and pass them correctly formatted towards the Interface.

Methods Database is the module where all the internal NESS available solving methods are stored. Other external solving methods may be available on CAS systems and are not stored here.

This component is a database system able to manage the access to the existing methods. The existing methods are available to the Expert and Solver in order to make reasoning or to solve the problem. The new methods, coming from the User Interface are stored and are also available to the Solver. The expert could not use the new methods in reasoning since it needs more data besides the method code.

A stored method may have associated two kinds of data: the code and reasoning data. If the code is enough for the method to be run, the reasoning data is important in the expertise. The method code is written is a dedicated language and will be interpreted at the Solver level. The reasoning data includes quantification of the method appropriation to be used to solve a problem with particular properties (see the section describing the possible problem properties).

The Problem module is split into three parts. The first part is maintaining the problem as the user introduced it. The problem remains available in the original form to be re-edited, saved, etc. The second part is intended to optimize the problem to be solved. Sometimes a problem may be simplified by replacing some variables computed function of others, when the problem can be decomposed in a linear part and a non-linear part.

The variables in the linear part will be expressed based on the variables in the other part. This is one case when optimization is needed. The third part detects the problem properties. This part contains routines performing the properties detection. The Expert uses the problem properties and methods properties in order to decide the suited methods for the problem.

The Solver is the part where the method is applied to the problem. A dedicated interpreter able to parse/execute the methods code interprets methods.
The expert supervised the entire process. It is automatically informed about the current iteration and internal events like floating overflow or division by zero. The process stops when either the Expert stops it or a solution was obtained.
The solver may use internal mathematical routines or external routines from CAS or mathematical libraries also figured in the figure 1. The routines in this category are high performance procedures for matrix operations or symbolic derivation routines.
The Expert is the component that copes with the intelligent aspects of the solving process rather than the mathematical aspects. The main functions of the Expert are:

1. Selecting the solving method based on problem properties and the properties of methods. The methods properties are obtained from the Methods Database module. The problem characteristics are coming from the Problem part. A matching mechanism is provided at the expert level in order to accomplish this task.
2. Providing all the initial data, if not supplied by the user, required for that particular method. We include here an initial iteration and some other parameters like maximum number of iterations.
3. Supervising the execution steps of the selected method. This means that some static data must be maintained into the expert, which allows the expert to reason about the current state of solution. This way, it can stop the current method, decide if the current solution is good enough or start a new method. A choice here could be to choose new initial data for the same method and restart it.
4. Benefiting from user indications; sometimes, the user may provide useful information about the system's characteristics and, eventually a kind of indications about how these can be used. The expert must be able to benefit from this data and to integrate new facts in its knowledge base.
5. Another expert's goal is to offer details to the user about the motivations to select a particular action. The explication are useful for advanced users which can decide if the expert reasoning is right or if it needs some adjustments.

4. Multi-Agent Model

In this section we present a general inter-agent communication model to be used in the NESS multi-agent system. The architecture of the agent system is presented in the figure 2.
The involved agent types are *Central Agent, Mediation Agents, Facilitator Agents* and Expert *Agents*.
We consider an expert agent having one or more capabilities (competencies) which can be used in behalf of other agents. A capability is associated with a task the agent may fulfill and has a central role in the overall architecture. Using the capability concept, an agent is represented in the environment by its name and its capabilities.

4.1. The Expert Agent

Essentially we distinguish between two main aspects: *communication* and *knowledge*. These aspects determine the layers of an agent: the communication layer and the expert layer as represented in the figure 2.

4.2. The Communication Layer

The need for communication resides in four aspects:

1. offer local data to other agents
2. request data from other agents
3. request a task execution from other agent
4. return answers to execution queries.

The agent environment uses the KQML (Knowledge Query Manipulation Language) [11] in order to transport the queries and answers between agents. This language offers the possibility to pass embedded messages in some particular language (KIF, Prolog, Lisp, CLIPS, etc) depending on the application. The user has also the choice to add a language translator at the expert level of an agent and can use different ``language speaking" agents. The speaking context for a conversation represents an ontology [8]. So, we mainly distinguish two ontologies: the *agent ontology* used in inter-agent communication and the *application specific ontology (ies),* used in the problem context.
The agents exchange data in an asynchronous manner. A synchronous approach may be implemented at the expert layer (if desired).
The communication layer provides a *mailbox* where the incoming requests are stored until retrieved for processing. The outgoing queries or answers are also placed in the mailbox.

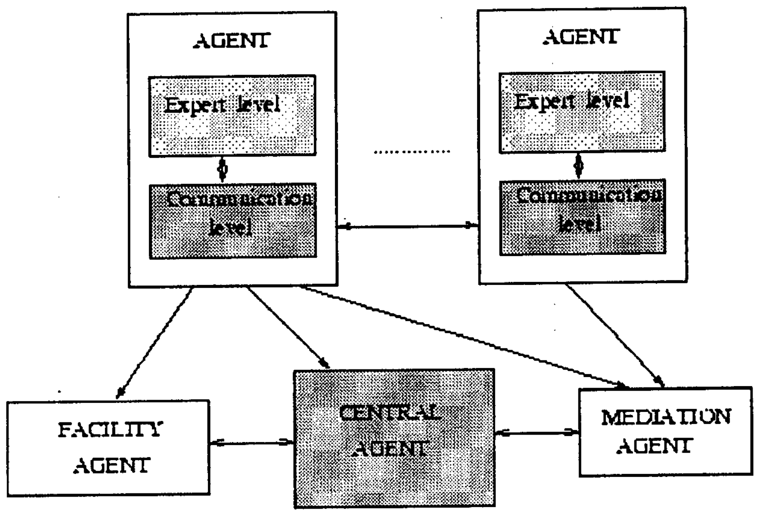


Figure 2: Multi-agent system structure

The mailbox structure is the only interface between the expert layer of the agent and the communication layer. The agent names are converted at this level in agents identifiers which are consequently used in communication exchanges. There are also maintained cache tables storing mappings between agents identifiers, names and capabilities, as long as the agent contact other agents to request capabilities execution. This way we can reduce communication expenses.

4.3. The Expert Layer

The reasoning process of the agent is entirely implemented at the expert level. The expert layer generates requests to the communication level in order to call other agent capabilities. The implementation of this level totally depends on application.

4.4. The Central Agent

The central agent has the role of ANS (Agent Name Server) retaining the mapping between agents' identifiers and agent names. Besides, the central agent retains the agents' capabilities. The central agent is able to provide agents with physical addresses when it is requested about the agents to perform a task. This agent uses the *agent ontology* in order to provide desired data.

In order to maintain the tables, every agent joining the system needs to register itself to the central agent and when an agent left the system it must signal the central agent, thus unregistering itself. The capability list for an agent is also provided at the registering time.

Once the registering fulfilled, every other registered agent has access to the agent capabilities.

This is an example of a conversation between an Agent and the Central Agent:

```
; agent registration
(evaluate :receiver ANS :sender
Agent :language KQML :ontology
agent :content (register :name
Agent :address gibon:2122
:capabilities multiply_two_matrix
inverse_matrix)))

;request for agent identifier
(evaluate :receiver ANS :sender
Agent :language KQML :ontology
agent :content (ask-address :name
Agent1))

;answer
(evaluate :receiver Agent :sender
ANS :language KQML :ontology agent
:content (tell-address :name
Agent1 :address gibon:2122
:capabilities (multiply_two_matrix
inverse_matrix)))
```

It is possible for an agent to ``die" before unregistering due to external conditions. To treat

this situation, from time to time, the central agent sends control messages to every registered agent in order to receive confirmations that the agent is alive or not. When no confirmation is received the agent is automatically unregistered.
Obviously, the central agent communication identifier must be available to all agents before their logging in.
Due to its role, the central agent is a critical point in the system. Anyhow, when no central agent is available, the agents may eventually benefits from their local caches when requesting capabilities from other agents, until the central agent is up again.

4.5. The Mediation Agent

The role of the mediation agent is strongly related to agent capabilities because this type of agent is intended to mediate inter-agent conflicts generated by having the same capabilities. The difference is made by problem restrictions. The mediation agent is able to process the problem restrictions or properties and to choose the appropriate agent to process the request.
Considering, for example, the problem of inverting a matrix. There are several methods to do that, depending of the matrix properties: symmetric or not, triangular or not and so on. The mediation agent (if it has the appropriate knowledge) decide which agent to propose, based on dialogs with all conflicting agents.
It is imperiously necessary for a mediation agent to be able to discuss with the involved agents and to be able to obtain problem specific properties in order to make a good choice.

4.6. The Facilitator (Utility) Agent

This type of agent is useful when implementing agents groups. Between a group of related agents it is often the case to exchange individual or broadcast information in order to maintain the consistency of some data or to keep trace of some common variables.
In order to implement this behavior every agent willing to offer information send an *advertise* message to the facilitator specifying which type of data it offers. The requests come to the facilitator, which then interrogates the adequate advertising agents building the response for the requesting agent.
The facilitator is also able to broadcast available data to all interested agents. The KQML language provides mechanisms for such a behavior.

5. A prototype design

We place our architecture in a UNIX environment, with the user interface agent able to run in any Java supporting environment. This is the only agent needing to be eventually ported. The rest of the agents are running as separate processes in a solving environment. The solving architecture accepts queries and establishes communications with the user interface agent.

5.1. Clips extension for communication

The communication acts of the CLIPS expert agents is entirely done through the KAPI agents environment developed at Enterprise Integration Technologies Communication. This environment is also able to provide communication mechanisms between agents programmed in LISP or C/C++.

5.2. Clips extension for task representation

Our task-oriented model is based on the task structure (3) and allows a reasoning and control knowledge explication at different levels of abstraction. A task is an elementary component of reasoning and represents a knowledge granularity level.
We tried to map the concept of tasks as described above on the CLIPS rule based knowledge system and its modularization and OOP concepts.
Using the CLIPS COOL facilities (OOP part of CLIPS), we have designed a task class, which includes precondition, post-condition, task-body and subtasks together with a method to activate the task.
This model of task allows us to simulate a backward chaining reasoning in the frame of forward chaining CLIPS inference engine. We have also designed a *task-monitor* entity whose purpose is to be a manager of all the existing tasks. This manager is on object of a dedicated class, which has as members a list with all the tasks, and a stack used by the monitor to activate in the right order the tasks.

6. Implementation Issues

One main issue is the portability aspect. We are strongly interested to create a system which can be used on most existing platforms. A solution of this problem is to provide portability only for the interface agent we talked about above. The solving environment may be placed in a dedicated place (platform) along with all the required environments such as symbolic computing ones, expert systems, DBMS's, etc.

At this moment the **Java** technology seems to offer all the required portability across systems and is ideal for developing user interfaces and we use it at the User Interface level. A big advantage is that there are some inter-agent communication systems already developed in Java, like JAT (Java Agent Template), available on the Internet.

We integrated the JAT environment with the KAPI agent environment (also available on Internet) in order to have an environment able to support both Java and non-Java agents communicating in the KQML language.

The design option allows many users accessing the NESS environment at the same time through WEB browsers or applet viewers from anywhere on the Internet.

Although this approach seems fine, a big issue is raised: how to keep trace of different sessions inside a single solving environment? This can be mainly solved in two ways: duplicating all the agents when a request comes in or implementing the solver in a manner allowing to maintain multiple data structures, one for each user session. The second alternative seems more attractive since it involves fewer resources to be consumed. Anyway, it requires complex multiplexing techniques at the solver level.

There are some other aspects to be accommodated, like the possibility to use different environments providing numerical or symbolical methods. Here we include the Computer Algebra Systems like Maple, Mathematica, or numerical libraries. The communication protocols must agree with the protocols supported by these systems. Open-Math seems to be a good choice, as an example of a mathematical object communication protocol because this is an emerging standard in the field. This type of protocols must be embedded in an envelope language that is largely used for inter-agent communication. An example is the KQML [11] language used for the inter-agent communication.

7. Conclusions

In this paper we describe a distributed multi-agent system architecture intended to solve distributed problems with a great degree of non-determinism. Every agent could run independently, but we have insisted on the social behavior.

We can point out some of the system qualities:

1. Portability across systems;
2. Friendly user interface;
3. The modularization that derives from distribution;
4. Control distribution but also the availability of some degree of centralized control which make the architecture more controllable;
5. The transparency of communications at the expert agents level;
6. The abstraction of agent's reasoning using task concept;
7. Error detection and recovering;
8. System evolution tracing

In our future work we will insist on finishing the existing modules and implementing the remaining tasks. A multiplexing agent intended to allow concurrent queries for NES problems must also be implemented. Another direction is to implement security mechanisms in order to establish better interactions between agents. Another extension we envisage is to adapt the EPODE environment for differential equation problems in a similar architecture and to integrate both systems (NESS and EPODE) in a common environment.

8. Acknowledgements

This research has been partially supported by Romanian Ministry of Education grant no. 12/1998 - D.G.I.S.C.S.U. and by World Bank grant no. 12623/98.

Bibliography

1. Brazier, F.M.T., Jonker, C.M., Treur, J., Modeling Project Coordination in a Multi-Agent Framework, IEEE Fifth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'96), 1996
2. Burg, B. and Arlabosse, F., ARCHON: Une platforme industrielle pour l'intelligence artificielle distribuee, Deuxiemes Journees Francophones IAD&SMA, 1994
3. Chandrasekaran, B., Design problem solving. A task analysis, AI Magazine, winter, 1990
4. Dennis, J. E., On the convergence of single-rank quassi-Newton methods, Math. Comp. vol. 24, 1970
5. Dennis, J. E. and More, J. J., Quassi-Newton methods, motivation and theory, SIAM J. Numer. Anal., vol. 19, no. 1, 1977.
6. Petcu, D. and Negru, V., Interactive system for Stiff computations and distributed computing, IMACS'98 Proceedings of advances in scientific computing and mathematical modelling, 1998

7. Gragg, W. B. and Tapia, R. A, Optimal error bounds for the Newton-Kantorovici theorem, SIAM J. Numer. Anal., vol. 11, no. 1, 1974
8. Gruber, T., Toward Principles for the Design of Ontologies Used for Knowledge Sharing, Stanford Knowledge Systems Laboratory, August 1993
9. Iglesias, C. A. and Gonsales, Z. S. C. and Velasco, J. R., MIX: A general purpose multi-agent architecture, LNCS 1069 - Distributed software agents and applications, 1994
10. Kamel, M.S. ,Ma, W.S and Enright, W.H, ODEXPERT: An expert system to select numerical solvers for initial value ODE systems, ACM Transaction on Mathematical Software, vol 19, no 1, 1993
11. Labrou, Y. and Tim Finin, T., A Proposal for a new KQML Specification, Technical Report no. TR CS-97-03, February 1997.
12. Maruster, S. Numerical methods in nonlinear equations solving, Editura Tehnica, Bucuresti, 1981
13. Negru, V. and Maruster, L., Non-linear equations systems solver, Artificial Intelligence: Methodology, Systems, Applications; Frontiers in Artificial Intelligence and Applications series, IOS Press, 1996
14. Casanova, H., Dongarra, J, Netsolve, a Network Server for Solving Computational Science Problems, The International Journal of Supercomputer Applications and High Performance Computing, vol. 11, no. 3, 1997
15. Ortega, J. M., Stability of difference equations and convergence of iterative processes, SIAM J. Numer. Anal., vol. 10, 1973
16. Talukdar, S. and Cardozo, E. and Podnar, G., Building large-scale software organizations, in Coupling symbolic and numeric computing in expert systems, Elsevier Science Pbs., 1988
17. Tong, S. S., Integration of symbolic and numerical methods for optimizing complex engineering systems, in Programming enviromnents for high-level scientific problem Solving, Elsevier Science Pbs.,1992
18. Trouilhet,S., Methode d'acquisition des connaissances sociales pour l'organisation d'une societe d'agents, Deuxiemes Journees Francophones IAD&SMA, 1994

AGENT FACTORY: TOWARDS SOCIAL ROBOTS

G.M.P. O'Hare, B.R. Duffy, R.W Collier, C.F.B. Rooney, R.P.S. O'Donoghue

PRISM Laboratory, Dept. of Computer Science, University College Dublin (UCD), Belfield, Dublin 4, Ireland

{Gregory.OHare, Brian.Duffy, Rem.Collier, Colm.Rooney, Ruadhan.ODonoghue}@ucd.ie

Abstract.
This paper advocates the application of multi-agent techniques in the realisation of social robotic behaviour. We present the Social Robot Architecture, which integrates the key elements of agent-hood and robotics in a coherent and systematic manner. This architecture seamlessly integrates real world robots, multi-agent development tools, and VRML visualisation tools into a coherent whole. Using these elements, we deliver a development environment, which facilitates rapid prototyping of social robot communities.

Keywords: multi-agent systems, agent architectures, agent-based robotics, co-ordination and collaboration, virtual reality.

1 Introduction

The primary concern of this paper is that of *Social Robotics*. Our research deals with the architecture whereby robot communities can engage in opportunistic collaborative behaviours in the solution of shared tasks.

This paper demonstrates the application of multi-agent techniques in the realisation of social behaviour between a group of autonomous mobile robots. To this end we commission *Agent Factory,* a software development environment that facilitates the rapid prototyping of Multi-Agent Systems (MAS). Agent Factory offers a conduit through which robot control can be governed by a deliberative agent architecture, specifically that of a Belief Desire and Intention (BDI) architecture. In addition, Agent Factory supports not only the creation of the social robotic community but also the subsequent experimentation and visualisation of their behaviour.

We integrate the key elements of agent-hood and mobile robotics in a coherent and systematic manner leading to the development and implementation of the *Social Robot Architecture.* This architecture seamlessly integrates real world robots, multi-agent development tools, and virtual reality visualisation mediums into a coherent whole. We therefore deliver a development environment, which facilitates rapid prototyping of social robot communities.

Section 2 offers an introduction to BDI architectures. Section 3 provides an overview of Agent Factory. An introduction to agent-based robotics is presented in section 4. Section 5 describes the *Social Robot Architecture* with section 6 animating its use. Discussion and conclusions are presented in section 7.

2 Belief Desire Intention (BDI)

Much research work has been commissioned on Multi-Agent Systems (MAS) and Distributed Artificial Intelligence (DAI) [BG88], [DLC89], [OJ95]. Specifically, competing agent architectures have been proposed in the literature. Two major architectural schools have emerged, namely those of *the reactive system school* and the *deliberative system school*. The former has predominated in the arena of autonomous mobile robot control. In this paper we advocate the synthesis of reactive and deliberative reasoning. In the delivery of computationally tractable models of deliberative reasoning, one approach that has gained wide acceptance is to represent the properties of an agent using mental attitudes such as *belief, desire,* and *intention*. In this terminology, an agent can be identified as having: a set of beliefs about its environment and about itself; a set of desires which are computational states which it wants to maintain, and a set of intentions which are computational states which the agent is trying to achieve. Multi-agent architectures that are based on these concepts are referred to as *BDI-architectures* (Belief-Desire-Intention) [RG91] [Jen93] [OJ96], and have recently been the subject of much theoretical research. Proponents of the BDI approach argue that the understanding of the dynamics of these mental attitudes and their intimate interdependencies, is crucial in achieving *rational* agent behaviour.

3 Agent Factory

3.1 What is Agent Factory?

In essence Agent Factory is a distributed environment for the rapid prototyping of intelligent agents. More complete descriptions of Agent Factory are presented elsewhere in the literature [OA95] [Col96] [OJ96] [OCC+98]. Agent Factory has been specified using the Vienna Development Method and realised using ObjectShare's implementation of Smalltalk-80, the VisualWorks integrated development environment.

Agent Factory is a member of the class of systems that embraces the BDI philosophy. The system offers an integrated toolset that supports the developer in the instantiation of generic *agent structures* that are subsequently utilised by a pre-packaged agent interpreter that delivers the BDI machinery. Other system tools support interface customisation and agent community visualisation. In creating an agent community three system components must be interwoven, those of *agents*, a *world* and a *scheduler*. The next section describes the high level architecture.

3.2 Schematic Functional Architecture

The development environment is based around Component Design Hierarchies (CDH) that are extensions of the standard Object Hierarchies in the OOP Paradigm. The development of specific agent communities requires the instantiation of particular designs from these design hierarchies. Three particular types of components have been identified as necessary for the development of agent communities: *agents*, *schedulers*, and *world interfaces*. Correspondingly, there is a CDH for each type. See figure 1.

The *agent* is the main computational unit of Agent Factory, it combines a series of attributes that represent and support the Mental State model of the agent, and a set of methods (the actuators). Agents are executed using an Agent Interpreter. This interpreter currently controls agent perception, and commitment management which considers the adoption, revision and realisation of Commitments.

The *scheduler* controls execution of the community, using an algorithm that exploits parallelism where possible.

Finally, the *world interface* acts as a medium between the problem domain, the community it is being developed for, and the other components of the Agent Factory system, as seen in figure 1.

4 Agent-Based Robotics

Initial research focused on the behaviour [Bro86]

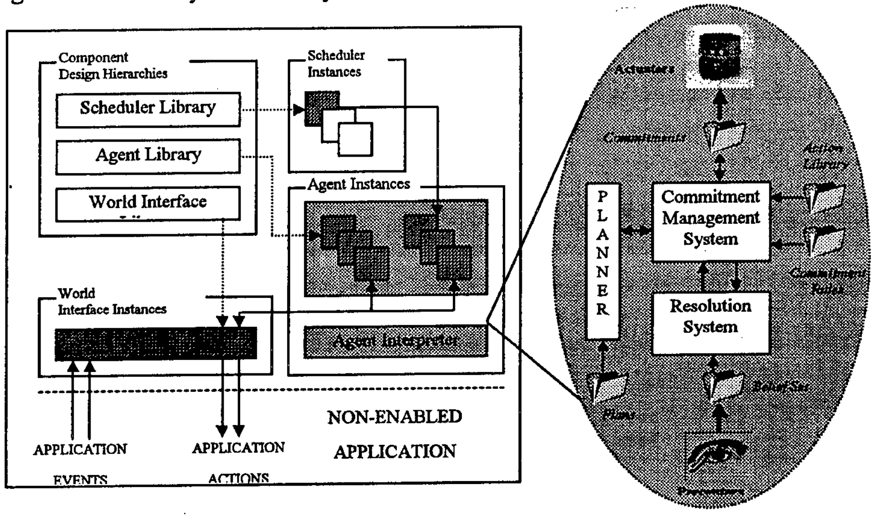


Figure 1: Agent Factory

[Ste94] and navigation problems of single robots, more recently the area of multiple robots has demanded considerable attention. There are numerous advantages in the use of multiple robots, these include *inter alia* distributed capabilities; parallelism; task and load distribution; increased functionality with minimal complexity. However, achieving coherent behaviour presents considerable challenges, none more acute than overcoming problems of co-ordination and interference. It seems clear that multi-agent techniques are amenable to transference to systems of multiple autonomous robots, in particular to addressing the problems of co-ordination and interference.

Initial work on agent based robotics emerged from cellular robotics where the robots had limited functionality and relied on swarm like intelligence to achieve their desired task, typically exhibiting *emergent* capabilities. Fukuda *et al* conducted research in the early stages of agent robots on both multi-robots in DARS (Distributed Autonomous Robotic System) [CF95] and reconfigurable robots within CEBOT (Cellular Robotic System) [Fuk+89].

Dudek *et al* in [DJM96] presents a taxonomy that classifies multi-agent systems according to communication, computational and other capabilities. Arkin and Balch [AB98] developed their reactive strategy for multi-agent co-operation with robots searching for trash objects, which they grasp and carry to wastebaskets. This research is concerned with the development of behaviours for formation maintenance in heterogeneous societies of mobile robots.

The RoboCup initiative [Rob95] provides a standard controlled environment for a team of multiple fast-moving robots to perform tasks, namely soccer playing, under dynamic real-time circumstances.

"The Robot World Cup Initiative (RoboCup) is an attempt to foster AI and intelligent robotics research by providing a standard problem where wide range of technologies can be integrated and examined"

The various technologies being researched include design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion. RoboCup also provides a *softer* transition from theory to reality through its standard environments.

Balch *et al* researches the impact of communication in reactive multiagent robotic systems in [BA94]. Balch later uses the soccer scenario for evaluating agents in terms of performance, policy convergence, and behavioural diversity [Bal97].

5 Social Robots Architecture

The motivation behind this research is to demonstrate, in a tangible form, the correspondence between systems of multiple agents and systems of multiple robots.

5.1 Social Robots

We introduce the term *social robots* here. Pre-existing research has been undertaken, most notably by groups at the Universities of Edinburgh and Reading, which interprets sociality from the perspective of communication, learning and human machine interaction. Researchers at the University of Essex, under the auspices of the EOS+ [EOS+] project use the techniques of Distributed Artificial Intelligence and artificial societies to study the emergence of human social complexity.

It is our conjecture that a distinction exists between *societal robotics and social robotics*. The former represents the integration of robotic entities into the human environment or society, while the latter deals specifically with the social empowerment of robots permitting opportunistic goal solution with fellow agents. This paper offers valuable insights into the delivery of, and experimentation with social robots.

5.2 Social Robot Architecture

The computational machinery needed to facilitate team building and collaborative behaviour is non trivial.

We describe the *Social Robot Architecture*, which goes some way toward achieving this through the judicious synthesis of the reactive model with that of the deliberative model.

The layered architecture (figure 2) has three fundamental elements: the deliberative level provided via Agent Factory, the robot level, and the environment level, which we consider in more detail in the subsequent sections. The control architecture offers basic reactive behaviours for reflex robot responses to unexpected or dangerous events. These constitute a set of primary *survival behaviours* for the robot. Goal oriented behaviours are delivered through the intentional agent structures of Agent Factory. Shen [SAC96] investigated a similar layered behaviours approach.

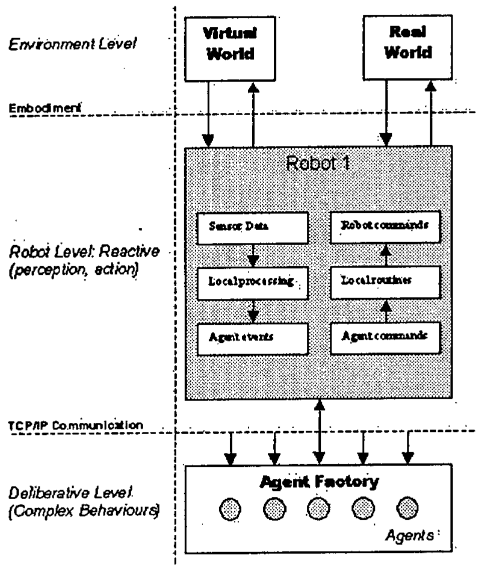


Figure 2: The Social Robot Architecture: The robot

Agent Factory supports inter-agent interaction (figure 3) enabling the resource bounded robot to usurp minimal computation on such activities, thus maximising resources for perception, reasoning, planning and action. Any process intensive action (i.e. planning) retards the real-time reflexive nature of the agent. Typically, more deliberative behaviour necessitates increased high level inter-robot communication, planning and deduction which could result in a loss of perception granularity.

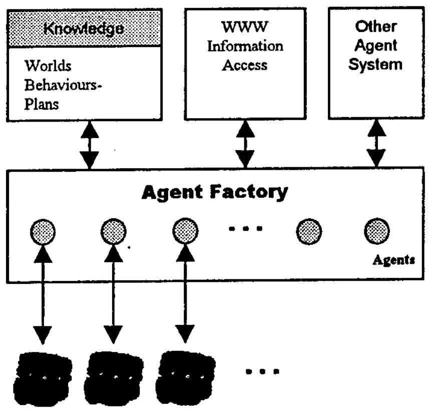


Figure 3: The Social Robot Architecture: Agent Factory

We now consider the functional components of the architecture

5.2.1 The Environment Level

Robots in our terminology may take the form of either that of a physical entity, specifically the Nomad Scout II from Nomad Technologies or a simulated entity, which may vary, contingent upon the visualisation medium. Robots are situated in the world and exhibit behaviour, based upon degrees of perception, action and interaction. Three such physical robots Aengus, Bodan and Bui[1] are available for experimentation. An example of such a world is given in section 7. Embodiment, as indicated by [Dre79] [LJ80], is an important element of the architecture for the realisation, implementation and testing of these behaviours.

This robot world has been mirrored in the real world and also, for the visualisation tool, in a VRML world. While the VRML world does not provide any feedback to Agent Factory, it constitutes a powerful and compelling visualisation tool for the development and experimentation of robot behaviour experiments. It also facilitates the ease of development of behavioural models within a dual observation medium providing analysis, recordability and alternate perspectives via both reality and a three dimensional metaphor.

5.2.2 The Robot Level

A series of fundamental behaviours are implemented at this level. They provide the basic *survival kit* of the robot necessary for such dynamic unpredictable environments. These *Survival Behaviours* include such behaviours as `avoid_obstacle`, `stop`, and `retreat`.

The sensory information received from the ultrasound and bumper sensor rings is processed at this level, resulting in clear agent-events being generated and communicated to Agent Factory.

5.2.3 The Deliberative Level: Agent Factory

The deliberative layer provides the deliberative machinery. This is achieved through the BDI architecture described in section 3. Beliefs are generated based on its belief set, and are updated with the receipt of new agent states or events from each robot. Our agents interact via an Agent Communication Language (ACL) based upon Speech Act Theory [Aus69]. The language, *Teanga*[2], is dealt with in more detail in section 5.3.

Agent events form the basic catalysts for robot communication and result in information being sent to the Agent Factory high-level controllers (agents).

[1] Aengus, Bodan and Bui are ancient characters in Irish legend and their graves can be found at Bru na Boinne, Ireland.

[2] Teanga is Irish for "language" and interestingly is an anagram for a agent.

Raw sensory data could be sent straight to Agent Factory however, this requires further processing to firstly filter and thus achieve attention focusing and to deliver the added value needed by our agents. Rather than burden Agent Factory with this low level processing, this is undertaken at the robot level, distilling the data down to a more useful form.

A scheduler in Agent Factory allots *time slices* to individual agents. As an agent enters its allotted time slice, it starts by gathering *perceptions*. Local robot processing filters these, generating a set of agent events that have occurred since the preceding time slice. The perception process within Agent Factory deals with converting these perceptions into beliefs and adding them to the belief set providing the agent with an up to date model of its current *perceived* situation. This situation is then analysed and the agents' commitments are updated. Those commitments that the agent made for realisation in the current time slice are analysed and, where possible, `agent_commands` are dispatched to the robot.

By way of example a team of robots may have a goal of mapping their environment. Decomposing this problem further, the individual robots might have to map constituent rooms, which in turn are made up of walls and a door. By analysing the data gathered from the sonar's these components might be identified. If a wall is found the robot then generates an agent event of the form, `wall_from(X,Y,Xi,Yi)`. Upon receipt the robot agent will revise it's belief set and accordingly add the belief `Bel(wall_from(X,Y,Xi,Yi))`.

5.3 Teánga

Our agents interact via an Agent Communication Language (ACL), called Teánga, which is based upon Speech Act Theory [Aus69]. Teánga consists of 4 basic categories of *communicative acts* (messages): informatives, directives, commissives and declarations. This categorisation was developed from a classification of performatives proposed by [Searle76]. Within each category there are more specialised types of communicative acts, e.g. `drop_commitment`. The language is designed as a carrier for an application-dependent content language. We do, however, place some constraints on prospective content languages. In the context of the Social Robot Architecture the content language must be able to allow, amongst other things, the representation of actions (including speech acts) and their status e.g. done, doing; and an agent's mental states, e.g. beliefs and commitments.

One of the main reasons why we chose to develop our own ACL rather than work with an established language such as KQML [LF94] or FIPA's ACL [FIPA97] is that our language should be compositional. We wish to be able to support nested speech acts (and speech acts contained inside composite actions, e.g. plans). To implement a nested communicative act in KQML requires that the content be a KQML message. However, in KQML the content language is independent and there is therefore no content checking. *"A disadvantage of content independence is that it prevents the content from being checked for compatibility with the speech act type"* [CL95].

The communicative acts within the real world are sent via wireless Ethernet.

5.4 Reality and Virtuality

One of the key tenants of our research has been the provision of multiple views of the operation of the same robot system. Consequently, the environment level provides two system views. The primary view provides a physical perspective of the Nomad Scout II's navigating the *robot world*. We supplement this with an abstract view to support behaviour generation and testing. The secondary view is a virtual reality perspective provided via the Agent Factory Visualiser which deliveres a 3-D VRML world via the Internet (figure 4).

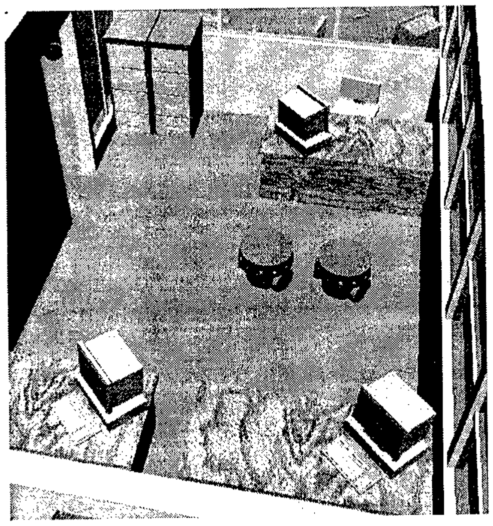


Figure 4: Virtual reality view of the IMPACT research room

Herein we harness the advantages of using virtual environments, by directly relating virtuality and reality in a seamless manner. Such alternate views permit multiple views, information hiding and abstraction, system interaction, and behaviour scrutiny via snapshots and recordings.

Our architecture supports the specification and invocation of robot experiments via a Java Internet interface. Each world is activated and their views synchronised.

5.5 *Incremental functionality*

In order to implement the complex notion of social robotics, we have broken the levels of behavioural complexity into incremental functionality as shown in figure 5.

As robot control stems from simple `move` and `turn` commands to complex notions of social robotics, this allows for systematic task decomposition and development leading to individual robot behaviours and extending to multiple robot behaviours.

A library of robot behaviours exists which adopts a subsumption model [Bro86]. Higher layers provide increasing complexity and subsume lower level functionality (see figure 5). Reactive or reflex survival behaviours are implemented at the robot level with more complex behaviours defined within Agent Factory. Each robot therefore has a degree of autonomy, which is independent of any inter-robot communication, and important in the realisation of robust complex robot behaviours.

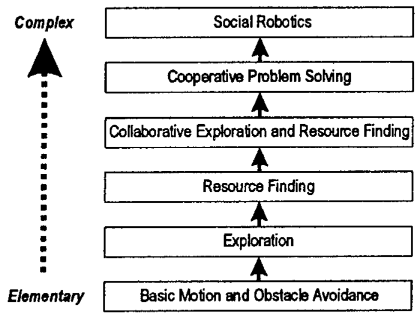

Complex
Social Robotics
Cooperative Problem Solving
Collaborative Exploration and Resource Finding
Resource Finding
Exploration
Elementary
Basic Motion and Obstacle Avoidance

Figure 5: Layered social behaviours

5.5 *The Virtual Reality Workbench*

Detailed programmes of robot tests have been performed and are characterised in the subsequent section. As a part of this research however we also wished to provide a shared global resource available and usable across the internet which supported distant robot experimentation.

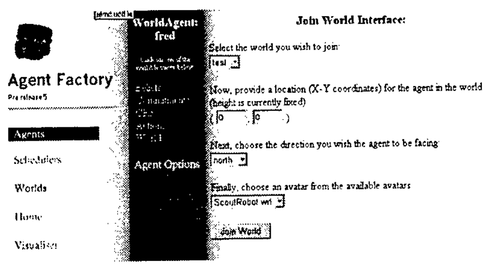
This facility, the *Virtual Robotic Workbench* , provides a medium through which researchers can design robot experiments remotely and without recourse to the purchase of expensive robotic entities in the first instance.

Researchers can articulate their experiments across a Java interface whereby they tune certain key workbench parameters, namely:

- The Customisation of the Environment
  - The Number of Robots;
  - The world within which they are to be situated;
- The Customisation of the Robots
  - Their Name;
  - The visualisation avatar associated with each robot;
  - Mapping virtual robots to physical robots;
  - The behaviours ascribed to a given robot;
  - Their Initial Location within the selected world;
- The Task
  - The selection of the shared goal;

Figure 6 depicts the nature of the Java interface by which these are specified Once the experiment has been crafted the subsequent activation results in the observable behaviour of the robots across any suitably configured web browser. The behaviour of the robots is governed by the social robotic architecture with the higher level functions directly furnished through Agent Factory. As such the virtual experimental results and the navigation and behaviour of the robotic avatars can be seen by utilising the Agent Factory Visualiser. Detailed treatment of the Agent Factory Visualiser is presented elsewhere in the literature [OCC+98]. Agent Factory generated events governing agent behaviour are dispatched simultaneously to both the robot controller, which activates the motors on the individual physical robots, and a proxy server, which, via the External Authoring Interface (EAI) of VRML, updates the world view accordingly, resulting in the movement of the virtual robot. The difficulty is in trying to achieve synchronisation between these two views[3]. Figure 7 illustrates the Agent Factory Visualiser and gives the reader a feel for the potential which the virtual robotic workbench offers.

[3] Clearly only in certain limited circumstances will it be possible for a corresponding physical experiment to be conducted at the same time. It is possible to provide a richer palette of virtual worlds for experimentation.



Join World Interface:

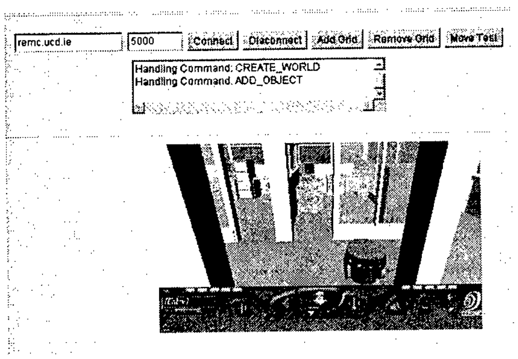
Select the world you wish to join: test

Now, provide a location (X-Y coordinates) for the agent in the world (height is currently fixed)
(0 , 0)

Next, choose the direction you wish the agent to be facing: north

Finally, choose an avatar from the available avatars: ScoutRobot wrl

Figure 6: The Agent Factory World Interface



```
Handling Command: CREATE_WORLD
Handling Command: ADD_OBJECT
```

Figure 7: The Agent Factory Visualisor

The benefits of the application of virtual reality to robotic experimentation are many-fold. Perhaps, the most obvious benefit is that it allows for remote behaviour observation with multiple perspectives. There is also the fact that any website that provides remote real-robot experiments is constrained by the number of physical robots available. The robots may be in use or offline. Our system can utilise simulated robots if the real-robots are unavailable thereby providing the user with an accurate 3-D visualisation of the experiment. Finally, on a more logistical note there is maximum resource utilisation through the participation of different research groups.

6 Application

With the building of a virtual reality representation of the PRISM research Laboratory and the robots visualised in this environment (as shown in figure 8), we have a testbed within which to demonstrate that a constructive link between reality and virtuality can be achieved.

This link is created and maintained through the Agent Factory World Interface (see figure 6).

This web-based interface allows the user to create, view and manipulate agents. These agents may then be situated in worlds and linked with real robots. A world can therefore exist in two environments: physically and in Virtual Reality (VR) as shown in figure 9.

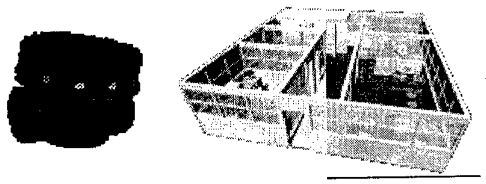


Figure 8: The virtual robot and PRISM Research Laboratory

Robot experiments are characterised by firstly selecting a world, subsequently situating robot(s) in this world, and finally ascribing behaviours to these robots.

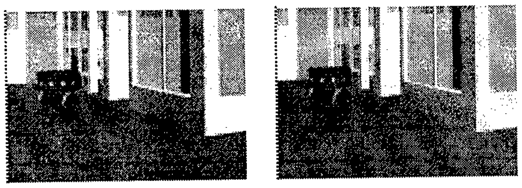


Figure 9: The virtual reality and physical reality in parallel

6.1 The Waltz

One such experiment, which exercises the degree of architectural and behavioural functionality required to demonstrate the applicability of the *social robotics architecture,* is that of a waltz duet. The specific waltz is the "Box Waltz" which consists of a simple square shape repetitive motion. The objective is for the two robots, Aengus and Bui to perform the waltzing behaviour side by side in a space restricted environment.

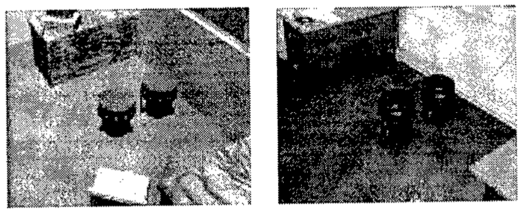


Figure 10: The robot duo performing the waltz and visualised in both reality and virtual reality

Figure 10 shows both a real and virtual perspective of the experiment and the synchronous motion of the robots.

Figure 11 demonstrates the robot trajectories while performing the waltz, clearly showing the behaviour modification when one robot encounters an unexpected obstacle or wall while also utilising the concepts of collaboration and communication to maintain a co-ordinated behaviour.

This experiment clearly demonstrates a robust system which, through the fusion of reactive survival behaviours and high-level deliberation, communication and co-ordination, can achieve complex goals in a real-time, real-world environment.

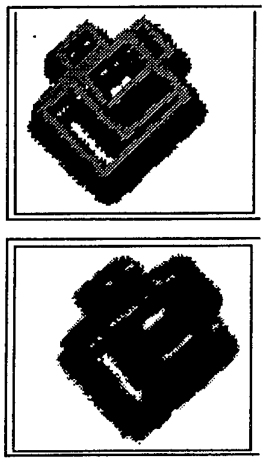


Figure 11: The two robot trajectories of the performed behaviour in a rectangular room

7 Discussion and Conclusions

This research has introduced *Social Robots* and an accompanying *Social Robot Architecture*. The architecture provides firstly, a powerful robot control mechanism, which integrates reactive and deliberative features and secondarily, a rich visualisation medium, which offers seamless, contrasting, yet consistent views of social robotic behaviour. Subsequently *social robot* experiments may be observed and analysed through these mediums.

With the breakdown of robot behavioural complexity, this has enabled the realisation of the social robotics concept via a series of developing stages of complexity and functionality.

To date, experimental evidence has demonstrated that *constructive* and *coherent* collaboration is achievable in restricted task domains. We are confident that our social agent architecture is scalable and ongoing research seeks to demonstrate its applicability to different real-world problem domains.

9 Acknowledgements

We gratefully acknowledge the support of Enterprise Ireland through grant No.'s SC/96/621 COMEDY and SP/97/08 IMPACT (Initiative for Mobile Programmable Agent-based Computing Technologies).

10 References

[AB98] Arkin, R.C., Balch, T. Co-operative Multi-agent Robotic Systems", *in Artif. Intel. and Mobile Robots, MIT/AAAI Press, 1998*

[Aus69] Austin, J. L., *"How To Do Things With Words"*, Oxford University Press, 1969.

[BA94] Balch, T., Arkin, R.C. "Communication in reactive multiagent robotic systems", *Autonomous Agents, 1, P1-25, 1994.*

[Bal97] Balch, T. "Learning Roles: Behavioural diversity in robot teams", AAAI workshop on Multi-Agent Learning, 1997.

[BD97] Billard, A., Dautenhahn, K. "Grounding communication in situated, social robots, *TIMR, Manchester 1997*

[BG88] Bond, A.H., Gasser, L. eds. "Readings in Distributed Artificial Intelligence", *San Mateo, CA., 1988.*

[Bro86] Brooks, R. A. "A Robust Layered Control System for a Mobile Robot,", *IEEE Jour. Rob. and Autom., 2(1) 1986*

[CF+95] Cai, A, Fukuda, T., et al. "Hierarchical Control Architecture for Cellular Robotic System – Simulation and Experiments", *IEEE Int. Conf. On Robotics and Automation, 1995,*

[Col96] Collier, R. "The realisation of Agent Factory: An environment for the rapid prototyping of intelligent agents", *M.Phil., Univ. of Manchester, 1996.*

[CL95] Philip R. Cohen & Hector J. Levesque, "Communicative Actions for Artificial Agents". *Proceedings of the International Conference on Multi-Agent Systems, AAAI Press, San Francisco, June, 1995.*

[DJM+96] Dudek, G., Jenkin, M., Milios, E., and Wilkes, D., "A Taxonomy for multi-agent robotics", *Autonomous Robots, Vol 3, 375-397, 1996*

[DLC89] Durfee, E.H., Lesser, V.R., Corkhill, D.D., "Trends in co-operative distributed problem solving", *IEEE: Knowl. Data Eng. 11(1), 63-8, 1989*

[Dre79] Dreyfus, H., What Computers Can't Do. *Harper, 1979.*

[EOS+] http://cswww.essex.ac.uk/ Research/AI/DAI/

[FIPA97] FIPA 97 Specification Part 2: Agent Communication Language. Foundation for Intelligent Physical Agents, 1997, http://drogo.cselt.stet.it /ufv/leonardo/fipa/spec/

[Fuk+89] Fukuda, T., et al. "Structure Decision for Self Organising Robots Based on Cell Structures", *IEEE: Rob. & Autom., Scottsdale Arizona, 1989,*

[Jen93] Jennings, N.R. "Specification and implementation of a Belief-Desire joint intention architecture for collaborative problem solving" *Int. Jour. of Intel. and Co-op. Info. Sys. Vol. II no3, 1993.*

[LF94] Yannis Labrou & Tim Finin, "A Semantics Approach for KQML- A General Purpose Communication Language for Software Agents". 1994,

[OA95] O'Hare, G.M.P. and Abbas, S., "Commitment Manipulation within Agent Factory", *Proc. of Decent. Intel. & MAS, Cracow, Poland, 1995*

[OJ96] O'Hare, G.M.P. and Jennings, N.R. (Editors.), Foundations of Distributed Artificial Intelligence, *Wiley Interscience Pub., New York, 1996, 296 pages. ISBN 0-471-00675*

[OCC+98] O'Hare, G.M.P., Collier, R., Conlon, J. and Abbas, S., "Agent Factory: An Environment for Constructing and Visualising Agent Communities", *Proc. AICS98., 1998*

[LJ80] Lakoff, G. Johnson, M.. "Metaphors We Live By", *University of Chicago Press, 1980.*

[RG91] Rao, A.S. and Georgeff, M.P., "Modelling Rational Agents within a BDI Architecture", *Prin. of Knowl. Rep. & Reas., San Mateo, CA., 1991*

[Rob95] The RoboCup Official web site http://www.robocup.v.kinotrope.co.jp

[SAC96] Shen, W., "YODA, Young Observant Discovery Agent",Proc. AAAI-96, Portland, Oregon.

[Searle76] J. R. Searle, , "The Classification of Illocutionary Acts". *Language in Society, 5, pp 1-24, 1976*

[Ste94] Steels, L., "Building Agents with Autonomous Behavior Systems", The 'artificial life' route to 'artificial intelligence'. Building situated embodied agents. Lawrence Erlbaum Associates, New Haven. 1994

MULTI-CRITERIA NEGOTIATION IN MULTI-AGENT SYSTEMS

Eugénio de Oliveira[1], José Manuel Fonseca[2], Adolfo Steiger-Garção[2]

[1] *FEUP - Faculdade de Engenharia da Universidade do Porto*
Rua dos Bragas - Porto – Portugal
e–mail: eco@garfield.fe.up.pt
[2] *FCT/UNL - Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa*
2825 Monte de Caparica – Portugal
e-mail: jmf,asg@uninova.pt

Abstract

Negotiation is a kind of decision making where two or more parties jointly search a space of solutions with the goal of reaching a consensus. This search is usually a single dimensional process where the different agents involved on the negotiation process mutually adapt their price offers in order to reach their goals (to sell or to buy a good). However, this process is inflexible and often unrealistic because more than one single issue has to be negotiated in order to find out an adequate solution. In this paper we present a multi-issue negotiation protocol for one-buyer-to-many-sellers interaction. The presentation is also enhanced with an application example that illustrates the negotiation process and shows how it can be useful both as a product brokering and as a market brokering tool.

Keywords: *Negotiation, Multi-issue negotiation, Electronic Commerce.*

1. Introduction

Most of the well known negotiation protocols deal with a single dimension (usually the price). Moreover, M. Wellman [Wel99] also says that the preferences reduction to the price single figure is what characterises the market. However, this seems to be a bit simplistic and unrealistic as well in the electronic commerce framework.

In this paper we are going to focus our attention on the kind of negotiation that takes place in an electronic market involving one buyer and many merchants that are potential sellers. In fact, we will use the traditional Consumer Buying Behaviour Model (CBB) that is also used and enhanced to accommodate software agents concepts as it is done by [Gut98a]. CBB encompasses six main stages: Need Identification, Product Brokering, Merchant Brokering, Negotiation, Purchase and Delivery, Negotiation, Purchase and Delivery and finally Service and Evaluation. Currently, Electronic Commerce agent mediated systems try to automate, at most, three of these stages (Product and Merchant Brokering and Negotiation). Here in this paper we are most concerned with the use of Software Agents for a rich (multi-dimensional) and flexible (adaptive) design of the Negotiation phase protocol. However, as we will see later, this negotiation protocol can also be seen as a product brokering stage once the buying agent can also use it as a market search process.

Negotiation is a kind of decision making where two or more parties jointly search a space of solutions with the goal of reaching a consensus [Ros94]. Some authors [Vul98] advocate the use of auction protocols for agent-mediated EC, arguing that they are widely recognised by economists [Moo92] as the most efficient way of resolving one-to-many negotiations.

Other authors point out the limitations of auctions' protocols and seek for more flexible negotiation models [Gut98b]. They stress out that online auctions are in fact less efficient and more hostile than it would be desired. For example, the winner's curse, i.e. the winner pays more than the real value of the product, seems to be a consequence of auctions and, therefore, they propose more co-operative Multi-attribute decision analysis tools and negotiation protocols using distributed constraint satisfaction policies to support it.

However, both lines of research are aware of the need of enhancing simple, price-based auction mechanisms, transforming them into new protocols encompassing multi-dimensional issues to be negotiated among the market participants.

This kind of negotiation is no longer of a win-lose type (zero-sum game on the game theory taxonomy) and becomes one of the win-win type (non-zero sum game) allowing agents to co-operate when negotiating over multiple dimensions. Therefore, co-operative negotiations can be described as the decision making process of resolving a conflict involving two or more parties over multiple independent, but non-mutually exclusive goals [Lew97]. Another important point to take into account in which agents negotiation in the electronic commerce framework is concerned, is the need for real adaptive agents that can learn with past experience in several different ways in order to become more effective in the market. Interesting work in multi-issue negotiation is being carried out by Nick Jennings [Vul98; Sie97]. However, the focus of their work refers to one-to-one negotiation while our work is dedicated to one-to-many negotiation once this is the typical situation in on-line shopping.

2. Issues on Multi-criteria negotiation

It is not crucial whether we use multi-lateral negotiations or auction protocols for agent-mediated electronic commerce as far as we enhance the negotiation protocol capabilities with multi-attribute bids and appropriate evaluation functions. Moreover, and according to [Vul98] and [Moo92], once the only auction protocols relying on agents playing their dominant strategies are the English and the Vickrey auctions, it seems reasonable to try to accommodate the new co-operating negotiating facilities along the multi-dimensional buyers' preferences to the referred auction protocols. Vulkan and Jennings introduce an extension to the English auction protocol by which all the service provider agents have a preliminary negotiation stage ("pre-auction") to find out the best proposal which is, afterwards, proposed to the service seeking agents for negotiation (or take-it-or-leave-it). They claim that in doing so, they maximise the worst-case outcome for the buyers (always restricting the auctions to private and not common values, i.e. not allowing goods re-selling). However, the referred authors deal with the agents' preferences as if they were just quantitative parameters through the definition of constraints on parameters.

Multi-criteria negotiation is extremely useful in a large number of situations. Situations where more than one single parameter is simultaneously taking part in the same negotiation are common in real commercial scenarios where buyers are confronted with several different appealing proposals.

In single-criteria negotiation the value of a parameter is negotiated with the different negotiators trying to defeat their opponents by offering a better proposal. The preferences of the buyer can be either implicit (the lower is the price the better for the buyer) or explicit with the buyer publicising its preference. This process can be done in a single round, where the competitors have one single chance to offer and no offer revision is allowed[1], or in multiple rounds where agents can revise their offers in face of the other competitors offers[2].

The first difficulty in multi-criteria negotiation is the buyers' preference definition. If we want to build a generic system, where no global, common knowledge is available, the buyer must make explicit his/her preferences in order to allow the potential sellers to calculate their best proposals taking also into account the buyer's point of view. While defining the buyer's preferences we must distinguish between discrete and continuous parameters. When the negotiation involves a discrete parameter it can be either a scalar or a non-scalar value. If it is a non-scalar value, the announcer (buyer agent) must specify either all the admissible values or a relationship over that parameter (higher than, lesser than, ...). Whenever the parameter's value is of a scalar type, the announcer can also specify intervals of admissible values. When dealing with continuous values, the announcer must specify specific values or intervals of acceptable values.

The second main difficulty in using multi-criteria negotiation is the mutual evaluation by both the buyer and sellers of each possible bid value. Because this evaluation must be done according to the buyer's criteria, the buyer must communicate his evaluation function to all the potential sellers. In order to allow the comparison of different possible bids including different parameters (due to different sellers' interests and possibilities) that cannot, in principle, be directly compared, the adopted solution is the weighting of the different possible values. In addition to the specification of the admitted values or intervals of values, the buyer must also specify the weight he/her attaches to each possible value. When intervals of values are defined, an evaluation for each interval must also be specified. Alternatively, an initial and a final value for each interval can be defined. In this case, the evaluation of each intermediate value will be calculated by linear interpolation. For continuous or discrete scalar values the upper limit for the interval can be considered as infinite as well as the lower limit can be considered as minus infinite.

[1] Vickrey auction and First-price sealed bid auction are examples of single-round protocols.

[2] The English and the Dutch auctions are probably the most popular examples of multiple-round auctions [San96].

3. A multi-issue negotiation protocol

The new protocol for multi-criteria negotiation we are here presenting is a modified version of the well known English auction. The buyer will start a negotiation session by announcing his request together with the list of the accepted values and correspondent evaluations. This announcement is then broadcasted to all the potentially interested agents. These receiving agents (sellers) evaluate the request and find their best answer (bid) for it. In this process, the sellers must weight two often antagonistic or, at least, non-coincident interests – their own interest as sellers and the buyer's interest. In order to maximise their profits, the sellers will produce bids that are not their own best bid (according to their own evaluation function), but those which they believe would be the best from the buyer's point of view. The negotiation process will force the sellers to move from an ambitious perspective (trying to maximise their own utility instead of the one from the buyer) to a non ambitious one (reducing their own utility in order to increase the buyer's utility) as the negotiation progresses. The process of finding a maximum for the linear combination of two non-linear generic functions such as both the buyer's and sellers' evaluation functions is a complex optimisation problem where genetic algorithms could be used to find out the optimal bid value according to the current constraints. The best solution found by each seller will then be considered as its first bid. Once the buyer has received all the bids (or a predefined timeout has expired) the bids are ranked and the best bid is selected. If the buyer, in face of the received bids, decides to change its own evaluation criteria, the evaluation parameters must be resent to all the sellers and the negotiation must be restarted. Otherwise, the best solution achieved is sent to all the sellers and a new negotiation round starts. In each round, the best bid is used as a reference bid. Only those bids that are better are accepted. If a seller sends out a bid with a lower buyer's evaluation, it is disqualified and eliminated from the seller's pool. If a seller has no chance to improve the best bid so far, its normal procedure is to quit the negotiation process explicitly by sending out to the buyer an adequate message. The negotiation process will finish when all but one seller quit or are disqualified.

The evaluation function used both by buyers and sellers is the following:

$$E(v) = \sum_{i=1}^{n} w_i . c(v_i)$$

$$c(v_i) = \begin{cases} \textit{if } p_i \textit{ discrete} \rightarrow T[v_i] \\ \textit{if } p_i \textit{ continuous} \rightarrow \dfrac{v_i - \min_i}{\max_i - \min_i} . (T[\max_i] - T[\min_i]) + T[\min_i] \end{cases}$$

where:

n – number of parameters
T[v_i] –user defined score to value v_i of parameter i
p_i – type of parameter i
Max_i – maximum value of parameter i
w_i – parameter i user defined weight
Min_i – minimum value of parameter i

The evaluation function $E(v_i)$ is a linear combination of the n parameters evaluation values represented as $c(v_i)$ and weighted by the weights w_i. The function $c(v_i)$ represents the score defined for the value c_i. If the parameter is discrete, such a result is a list of values, one for each possible value of the input parameter. If the parameter is continuous, the evaluation value is calculated by linear interpolation between the value ($T[min_i]$) corresponding to the minimum acceptable value for the parameter i (min_i) and the value ($T[max_i]$) corresponding to the maximum (max_i) acceptable value for the parameter i as defined by the user. The weights w_i are somehow redundant because the user could define different weights by correctly setting the evaluations $T[v_i]$. However, the explicit definition of the weights was considered to be much more understandable and natural for the user.

4. A negotiation example

In order to illustrate the concepts and protocol presented before, an application example is now introduced. The negotiation protocol is completely generic and can be applied to any (or almost any) field. To facilitate the reading of this paper, the automotive market - a very familiar scenario - was chosen. A situation where a user wants to buy a new car was simulated and the new protocol for negotiation is now applied, explained and discussed. After the user has selected a car in the category of "family car" he gets a pre-defined evaluation function that was created in order to correspond to the usual preferences of the typical "family car" buyer. However, the user can also freely customise this profile if it has different preferences. In this example, the weights considered in the evaluation function range from 0 to 100 and the score range from 0 to 20. However, any other scale can be used because only relative comparisons of values are made. The absolute value of the classifications is irrelevant. The weights referred above reflect the importance the user gives to each parameter.

	Brand			Number of doors				Price		ABS		AIRGBAG Driver		AIRGBAG Pass		Alarm		AC		Power steering	
Weight	100			20				100		20		20		20		5		50		30	
Type	D			D				C		D		D		D		D		D		D	
Value	Beta	amma	Delta	2	3	4	5	2000	6000	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Score	7	10	10	9	10	14	15	15	5	15	10	15	5	15	5	15	10	15	5	15	5

	Power windows		Metalic painting		Alloy wheels		Sunroof		Radio		Urban consumption		Capacity (cc)		Max. Power		Maximum Speed		Aceleration		Baggage capacity	
Weight	10		15		0		5		10		60		40		70		30		60		70	
Type	D		D		D		D		D		C		C		C		C		C		C	
Value	Y	N	Y	N	Y	N	Y	N	Y	N	4	12	1000	2000	50	150	150	250	5	25	200	1000
Score	15	5	12	10	11	10	14	12	8	6	20	2	15	16	5	15	10	12	20	0	5	20

Table 1 – User's preferences.

Classifications define the importance the user gives to each one of the discrete parameters' value or to each one of the limits of each continuous parameter domain. Classifications can also be used to express a low or a high importance to each parameter. By associating high classifications to the different values of a parameter (ex. 16-20) the user can also give it a higher importance in the final classification than what would be the case if using low classifications (ex. 6-10). This kind of redundancy was introduced in order to achieve a more natural representation of the user's preferences.

The preferences shown on Table 1 will be used both by the sellers to evaluate their offers from the buyer's point-of-view and by the buyer to evaluate the value of the different bids. The first line on table 1 is the *type* of each parameter. It can be either *discrete* (D) or *continuous* (C). If the type is continuous, the different accepted values for the parameter are given and under it, the correspondent classification is also given. If the parameter is continuous the two limit values (minimum and maximum) are defined and the classification for these two limits is presented. The evaluation function will use linear interpolation to evaluate intermediate values. The evaluation scale used was between 0 (minimum) and 20 (maximum). However, it is important to stress out that the scale used is irrelevant because the evaluation function will be used only for the sake of comparison between the different bids. This classification criterion will be appended to the announcement.

The same way as the values in table 1 reflect the evaluation criteria of the buyer, the different sellers will also have their own evaluation function to calculate the value of each possible bid. As well as the buyer's goal is to get the product that maximises his evaluation, the seller's goal will be to find the product and selling conditions that will maximise its satisfaction and simultaneously convince the buyer. The evaluation function of the sellers can be based on distinct parameters that the one of the buyer. For simplicity, we have used, in this example, the same evaluation function for all the sellers. However, each seller can use a completely different function with different parameters and corresponding classification. The evaluation function for the different sellers is presented on table 2.

Comparing table 2 with Table 1 it can be seen that only some evaluation parameters are common to both tables. New parameters like *Stock* (reflecting the number of units in seller's stock), *Model age* (how long is the car's model available in the market) and *Discount* (the percentage of discount relative to the ideal price) were now introduced. These new parameters, the absence of some of the previously defined and the different given values for the common parameters reflect the different perspectives the buyer and the sellers have for the same product. Of course, the sellers evaluation function is private to each seller and it is never revealed to the buyer or to any other seller.

The sellers, knowing both functions, can evaluate the products they want to sell, from both perspectives. To go on with our example, we will now consider three different sellers *Beta, Gamma* and *Delta* with different products to sell that can be of interest to the buyer who have made the previous announcement (see Table 3).

	ABS		AIRGBAG Driver		AIRGBAG Pass		Alarm		AC		Power steering		Power windows	
Weight	20		20		20		20		20		20		20	
Type	D		D		D		D		D		D		D	
Value	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Score	10	15	10	15	10	15	10	15	10	15	10	15	10	15

	Metalic painting		Alloy wheels		Sunroof		Stock		Model age		Max. Discount	
Weight	20		20		20		80		50		100	
Type	D		D		D		C		C		C	
Value	Y	N	Y	N	Y	N	0	100	0	6	0	25
Score	10	12	10	12	10	15	0	20	10	20	20	16

D – discrete parameter C – continuous parameter

Table 2 – Sellers evaluation function.

Brand	Model	Doors	Normal price	ABS	AIRBAG Cond	AIRBAG Pass	Alarm	Air cond	Power steering	Power windows	Metalic painting	Alloy weels	Sunroof	Radio	Urban consumption	Capacity (cc)	Max. Power	Max. Speed	Aceleration	Baggage capacity
Beta	Betamax SR	3	3098	n	s	n	n	n	s	s	n	s	s	n	9.3	1370	103	185	11.9	320
	Betamax AIRBP MP SR	3	3170	n	s	s	n	n	s	s	s	s	s	n	9.3	1370	103	185	11.9	320
	BetamaxAC AIRBP	3	3461	n	s	s	n	s	s	s	n	s	n	n	9.3	1370	103	185	11.9	320
	Betamax AC AIRBP MP	3	3491	n	s	s	n	s	s	s	s	s	n	n	9.3	1370	103	185	11.9	320
	Betamax AC SR AIRBP MP	3	3566	n	s	s	n	s	s	s	s	s	s	n	9.3	1370	103	185	11.9	320
Gamma	Gamex EL	5	3073	n	n	n	n	n	s	s	n	n	n	s	9.2	1370	80	170	13.9	380
	Gamex EL AIRBC AC	5	3505	s	s	n	n	s	s	s	n	n	n	s	9.2	1370	80	170	13.9	380
	Gamex EL AIRBC AIRBP AC	5	3545	s	s	s	n	s	s	s	n	n	n	s	9.2	1370	80	170	13.9	380
	Gamex EL MP	5	3113	n	n	n	n	n	s	s	s	n	n	s	9.2	1370	80	170	13.9	380
	Gamex EL AC MP	5	3338	n	n	n	n	s	s	s	s	n	n	s	9.2	1370	80	170	13.9	380
	Gamex EL AIRBC AIRBP AC MP	5	3585	s	s	s	n	s	s	s	s	n	n	s	9.2	1370	80	170	13.9	380
Delta	Delux	5	3085	n	s	n	n	n	s	s	n	n	n	n	10.1	1392	75	169	14.4	380
	Delux ABS	5	3185	s	s	n	n	n	s	s	n	n	n	n	10.1	1392	75	169	14.4	380
	Delux ABS AIRBP AC	5	3418	s	s	s	n	s	s	s	n	n	n	n	10.1	1392	75	169	14.4	380
	Delux ABS AIRBP AC AW	5	3533	s	s	s	n	s	s	s	s	s	n	n	10.1	1392	75	169	14.4	380

Table 3 – Sellers products description.

After received the announcement, the seller will bid with the solution that optimises its own criteria. If there is more than one solution with equal evaluation to the seller, it will decide for the one that is more valuable to the buyer. Once received the different solutions from all the sellers, the buyer selects the best proposal and communicates the evaluation of that proposal to all the sellers. In each negotiation round all the bids must have a higher evaluation than the best bid of the previous one plus some pre-defined threshold (in this example 0.1) unless it is their last bid. In this case any increment to the previous best is accepted. In Table 4 the buyer and seller evaluations are depicted. The minimal and maximal evaluations are due to the price factor. In this example this is the single parameter that can be continuously modified in between pre-defined limits set by the seller (set by the ideal price and maximum discount). Because the price has a direct influence in both the buyer and seller evaluations, when a car is proposed at its higher price, the maximum evaluation for the buyer and the minimum for the buyer will be obtained. When the minimum price is offered we will have the maximum for the buyer and the minimum for the seller. Any value between these two situations is possible depending on the percentage of discount adopted. A situation where a seller chooses to make a discount instead of moving to another product occurs in the third negotiation round of the example. Of course that this table can only be calculated in the vendor side. The buyer has no idea about what is the maximum discount the seller is able to do. Only the negotiation can force the seller to reduce its price moving from a zero discount offer to a reduced price offer. Anyway, the seller will always make the offer with the highest possible evaluation to himself. The price reduction will only be adopted if it is the best solution to the seller accordingly to its evaluation function.

In Table 4, the "Seller max evaluation" column corresponds to the seller's evaluation of each car at its maximum price and the "Seller min evaluation" column corresponds to the seller's evaluation at each car minimum price. In a similar way the "Buyer max evaluation" is the buyer's evaluation at the minimum possible price and the "Buyer min evaluation" column the buyer evaluation at the maximum possible price. This table is private to the sellers because the buyers do not know the minimum prices. The table is used to know the range of values each bid (car) can have, from the buyer's perspective, whenever the price (the only continuous parameter that the sellers can control in this example) is modified.

Lets now see how negotiation evolves from the announcement stage to the final assignment. The first step in the negotiation process is the announcement made by the buyer communicating the intention of buying a car and containing its evaluation function as described in table 1. In their first bid, the different vendors will bid with the solution that presents a better evaluation for themselves. As you can confirm in table 4, company *Beta* decides to bid with car *Betamax c/TA* (13.27 to *Beta*, 9.99 to the buyer), *Gamma* with *Gamex EL* (12.84 to *Gamma*, 10.01 to the buyer) and *Delta* with *Delux* (13.52 to *Delta*, 10.00 to the buyer).

Brand	Model	Seller max evaluation	Seller min evaluation	Buyer max evaluation	Buyer min evaluation
Beta	Betamax SR	13.27	10.60	10.14	9.99
	Betamax AIRBP MP SR	13.13	10.47	10.44	10.28
	BetamaxAC AIRBP	12.16	9.49	10.98	10.80
	Betamax AC AIRBP MP	12.24	9.58	11.01	10.83
	Betamax AC SR AIRBP MP	11.95	9.28	11.00	10.82
Gamma	Gamex EL	12.84	10.17	10.17	10.01
	Gamex EL AIRBC AC	12.46	9.79	11.13	10.95
	Gamex EL AIRBC AIRBP AC	12.55	9.89	11.39	11.21
	Gamex EL MP	12.68	10.01	10.20	10.04
	Gamex EL AC MP	12.81	10.15	10.81	10.64
	Gamex EL AIRBC AIRBP AC MP	11.97	9.30	11.42	11.24
Delta	Delux	13.52	11.74	10.10	10.00
	Delux ABS	13.37	11.59	10.21	10.10
	Delux ABS AIRBP AC	12.43	10.65	11.09	10.97
	Delux ABS AIRBP AC AW	12.11	10.33	11.10	10.98

Table 4 – Buyer and sellers evaluations.

Therefore, *Gamma EL* is the best offer and 10.11 is the price beat in the round (10.01 plus 0.1). In the second round *Gamma* does not bid because it has the best proposal. *Beta* proposes *Betamax* with passenger AIRBAG, open roof and metallic painting (13.13 to *Beta,* 10.28 to the buyer). *Delta* proposes *Deluxe* with ABS and a 2% discount in order to achieve an acceptable evaluation by the buyer (13.01 to *Delta*, 10.12 to the buyer). In the third round *Alfa* is leading. *Gamma* proposes now *Gamex EL* with air conditioned and metalised painting (12.81 to *Gamma*, 10.64 to the buyer) and *Delta* the *Deluxe* with ABS, passenger AIRBAG and the air conditioned (12.43 to *Delta*, 10.97 to the buyer). In the fourth round *Beta* is in serious difficulties to beat the third round best and does its best proposing *Betamax* with passenger AIRBAG, air conditioned, metalised painting and the lowest possible price (9.58 to *Beta*, 11.01 to the buyer). *Gamma* bids with *Gamex EL* with double AIRBAG and air conditioned (12.55 to *Gamma*, 11.21 to the buyer). This last offer is unbeatable by the other companies and both quit in the sixth and last negotiation round leaving *Gamma* as the negotiation winner.

5. Conclusions and future work

In this paper we have presented what we believe to be an adequate protocol for multi-issue, multi-vendor negotiation process. Inspired by the English-Auction case, the protocol now proposed supports multi-round negotiation of multi-issue evaluated transactions allowing the buyer agent to automatically find out the best of the proposed bids according to the buyer's interests. This solution is very well suited to the retail electronic market, once it promotes the buyer's proactiveness more effectively than the usual in current electronic auction systems. Instead of being up to the seller to promote the auction of the selling goods (or services) we are here proposing that the buyers can do it by launching buying announcements to which the sellers try to answer by making the best possible bids. Moreover, instead of negotiating just the price, as it is typical in most of the current systems, a large number of different issues that characterise the products should be negotiated. This means that if the buyer does not have a precise idea about what it really wants and is receptive to proposals including different features of a specific good (as it is often the case in real-life trading) he can specify large acceptance margins for the future bids and, through negotiation, to find out the most suitable product available in the market. Of course that broader parameter domains (due to relaxation) usually means more negotiation rounds and, as a consequence, more time consumption as well as more communication.

A limitation of the actual implementation is the restriction of the bids evaluation functions to linear functions. The introduction of Genetic Algorithms in the offer search process will allow the use of any evaluation function without any restrictions. Another important development will be the inclusion of an advanced protocol to support agents' coalitions formation. The introduction of mechanisms for coalition formation and intra-coalition negotiation identical to those presented in [Oli97] but now adapted to the multi-criteria negotiation, will be one of the next steps in the development of our multi-issue negotiation protocol.

Bibliography

[Gut98a] Guttman R., Maes P., Cooperative vs competitive Multi-agent negotiation in retail Electronic Commerce, in Proceedings of the 2nd Int. Workshop on Cooperative Information Agents CIA'98, Paris, 1998.

[Gut98b] Guttman, R. H., Moukas, A. G., and Maes, P., "Agent-Mediated Electronic Commerce: A Survey", Knowledge Engineering Review, June 1998.

[Lew97] R.Lewicki, D.Saunders, J.Minton, "Essentials of Negotiation", Irwin,97.

[Moo92] Moore J., Implementation, Contracts and Negotiation in Environments with complete information, in Advances in Economic Theory, V.1, Cambridge University Press, 1992.

[Oli97] Oliveira, E. d., Fonseca, J. M., and Garção, A. S., MACIV - A DAI Based Resource Management System, International Journal on Applied Artificial Intelligence, vol. 11, pp. 525-550, 1997.

[Ros94] Rosenschein, J. S. and Zlotkin, G., *Rules of Encounter*. MIT Press, 1994.

[San96] Sandholm, T. W., "Negotiation Among Self-Interested Computationally Limited Agents," PhD Thesys: University of Massachusetts at Amherst, 1996.

[Sie97] Carles Sierra, Jennings N., Noriega P., Parsons S. "A framework for Argumentation Based Negotiation" (ATAL97) Intelligent Agents IV (M. P.Singh, A. Rao and M. J. Wooldridge, eds). Springer-Verlag LNAI 1365 (1998) pp 177-192.

[Vul98] Vulkan, N. and Jennings, N. R., "Efficient Mechanisms for the Supply of Services in Multi-Agent Environments," presented at 1st Int Conf. on Information and Computation Economies, Charleston, South Carolina, 1998.

[Well99] Wellman,M. and Wurman, P. market-aware Agents for a Multiagent world to appear in Robotics and Autonomous Systems Journal 1999.

ON THE PROBLEM OF DETECTING SUBSUMPTIONS AND INCONSISTENCIES IN AGENTS KNOWLEDGE

Gerald S.Plesniewicz[1], Valery B.Tarassov[2]

[1]*Russian State University of Technology – MATI, Petrovka 27, Moscow 103767, Russia*

[2]*Bauman Moscow State Technical University, 2nd Baumanskaya st., 5; Moscow 107005, Russia*

E-mail: tar@srv-m.mpei.ac.ru

Abstract

We consider the problem of detecting subsumptions and inconsistencies in the knowledge available or obtained by intelligent agents. This problem is formulated in the context of agents modelling in terms of special CONCEPT language. For solving it we offer a method of compiling declarative knowledge into a set of production rules acting on fact bases. Then, knowledge inconsistency is detected whenever a fact base with contrary facts is reached under applying the set of productions

Keywords: *intelligent agents, multi-agent systems, knowledge inconsistency, default reasoning, nonclassical logics, concept-oriented language*

1. Introduction

Let us consider intelligent agents A_i, $i=1,\dots N$ which are capable to reason on the base of some priory knowledge K_i. Let $K_i(t)$ denote the knowledge that the agent Ai has at time t. (It is supposed that $K_i(0) = K_i$.) Here $K_i(t)$ is obtained from $K_i(t-1)$ by adding some new knowledge K and by simplifying the union $K \cup K_i(t-1)$, if possible. There are three ways of getting K:

- The agent A_i reasons on the basis of $K_i(t-1)$
- External (for N agents) sources of knowledge
- The agent A_j transfers (from time to time) a part of his knowledge $K_j(t-1)$ to the agent A_i.

However the following two questions arise in the context of maintaining the knowledge correct-ness and simplicity.

Q1: Is K subsumed or not by $K_i(t-1)$?

Q2: Is $K \cup K_i(t-1)$ consistent or inconsistent?

Recall that some knowledge K* is called inconsistent if there is no interpretation I such that K* is true under I; and the knowledge K* is subsu-med by the knowledge K** if for every inter-pretation I K* is true under I whenever K** is true under I. (It is clear that if K* is subsumed by K** then $K^* \cup K^{**}$ is equivalent to K**).

The questions Q1 and Q2 are related to local analysis of multi-agent systems. More delicate problems may be faced in global analysis. For example, the question Q3 is related to global analysis. To formulate it we take the following considerations.

Let $K(t) = \{K_i(t) \mid 1 \leq i \leq N\}$ be the knowledge that all N agents have by the time t; and let d(t) be the data introduced into the multi-agent system at the time t. Here a transition function Φ may be written in the form:

$K(t)=\Phi(d(t-1),K(t-1))$.

Q3: Does exist or not the time t and inputs d(i), $0 < i < t$, such that K(t) is inconsistent ?

In DAI the agents knowledge is often repre-sented by a set of sentences S with using a suitable formalism. It is well known that if the negation is allowed in this formalism then the following questions are equivalent: «Is S inconsistent»? and «Is S_1 subsumed by S_2»?

It is pointed out by Y.Shoham: «Most often, when people in AI use the term 'agent', they refer to an entity that functions continuously and autonomously in an environment in which other processes take place and other agents exist» [1]. Therefore, the most natural and efficient way to design intelligent multi-agent systems consists in using such facilities as active data models with autonomous objects. So the system DEGAS is an instance of such models [2]; more exactly, it is an advanced active data model with object au-tonomy and highly distributed control.

Each agent A can be represented in DEGAS system by an instance of the class *Agent* with an appropriate structure defined by its components, attributes, methods (functions), production rules, and its life cycle values. Here the methods and life cycle values specify the agents potential behaviour, whereas the execution of production rules (triggers) determine their actual behaviour. In particular, some of the triggers are used to manage knowledge exchange between agents.

However, in DEGAS system, it is possible to represent the agents knowledge only in the form of productions. Nevertheless, the productions languages provide rather low level tools for knowledge representation. To make easier the procedures of knowledge acquisition from domain experts and its representation, it seems to be more reasonable to use modelling languages of higher level which may be referred as cognitively more adequate. Among these languages the so-called concept-based (or concept-oriented) languages discussed in [3] are of primary concern. An example of such a language is CONCEPT [4] - an advanced concept-oriented language which is rather well adapted to specifying both the structure and behaviour of intelligent agents.

Firstly, CONCEPT is equipped with some facilities that are equivalent, in fact, to those of the DEGAS model. By using these facilities, the expert can design in a natural way a scheme for active databases with autonomous objects; this scheme specifies the structure and behaviour of the agents. Secondly, CONCEPT allows to spe-cify the agents knowledge in a declarative man-ner, i.e. in the form being often more suitable and natural than production sets. However, declara-tive sentences are not (directly) executable.

There are two ways of operational interpretation of declarative sentences: a) to use programs making inferences for such sentences; b) to translate these sentences into directly executable statements, for example, into productions. The lat-ter is more preferable because of the possibility to directly integrate compiled productions as trig-gers into active databases of the CONCEPT sys-tem.

In this paper we propose a technique of compi-ling declarative knowledge into productions, and show how the compiled productions can be used for detecting subsumptions and inconsistencies in the agents knowledge. Let us note that the use of productions leads to more efficient and regular computations for answering questions Q1 and Q2, and. in particular, question Q3. Indeed, the transition function Φ normally can be represen-ted by productions; therefore, it is rather easy to integrate this representation of Φ with the productions resulted from compiling a priory knowledge K(0).

For the sake of simplicity, we confine ourselves to the case of declarative knowledge represented in a concept-based language, namely, the CBRd language, which is an extension of the CBR language [5]. (The acronym stands for «Classes and Binary Relations»; the letter "d" says that the language includes defaults). Here CBRd is an abstraction of a certain subset of the CONCEPT; its kernel role is similar to that of propositional logic in predicate logic.

R e m a r k. In a sense, our method for detecting inconsistency looks like the analytic-tableaux method [6].

2.Some CONCEPT features

Denotational semantics of CONCEPT is based on a formal concept model which is close to the object class. The difference consists in the fact that here the concept includes points of reference and coreferentiality relation.

A *formal concept* C is a pair (U^C, Ext^C) where

- $U^C \subseteq U$, U is the universe (the set of possible object names); U^C is called the *universe of the concept C*
- Ext^C is a family $\{Ext^C_\gamma \mid \gamma \in \Gamma\}$ indexed by a set Γ of *points of reference*
- Every Ext^C_γ is the quotient $E^C_\gamma / \sim^C_\gamma$ of some subset $E^C_\gamma \subseteq U^C$ by some equivalence $\sim^C_\gamma$ called the coreferentiality relation. Here Ext^C_γ is called the *extension of C* at the *point of reference* γ.

Every name $e \in E^C_\gamma$ is (the name of*) an instan-ce* of C. If $e \sim e'$ then e and e' are coreferent; they both refer to the same object of a problem do-main.

A *concepts system* is a finite set of formal concepts with the same points of reference. So CONCEPT is a language for specifying the concepts systems. A text in CONCEPT called *conceptual scheme* consists of statement that

specify: a) concept universes U^C (*structural statements*);
b) concept extensions Ext^C_γ given independently from the points γ (*logical statements*);
c) pairs of concept extensions (Ext^C_γ, Ext^C_δ) with a pair $(\gamma, \delta) \in R$ (*transition statements*); here R is a binary relation in Γ (*accessibility relation*).
By means of structural statements one can specify: i) data types; ii) functions on the data ty-pes; iii) object-oriented structures with attributes, components, taxons, and inheritance mechanisms.
By means of logical statements one can define in a declarative form various constraints on concept extensions. The constraints select only those extensions that are allowed by the semantics of a given problem domain.
By means of transition statements one can des-cribe the «behaviour» of concepts extensions in response to points-of-reference changes. Specifi-cally, the productions are transition statements that specify the object dynamics.
A finite set of structural, logical and transition statements is called *conceptual scheme*. For example, on specifying in CONCEPT the structure of instances of the *Agent* concept we could write
Agent [Name:String,Type:String,State,LifeCycle,
LinksTo: Agent(*), Believes: Knowledge, Intends: Action, Informs: [Agent, Know-ledge] (*)],
Knowledge [Sentence(*) | Fact(*)], etc.
In CONCEPT there are many types of logical statements.The simplest statements have the form $A=B(\alpha)$, where α is an attribute expression interpreted as a requirement to instances of C. For example, the expression
Mobile Robot = Agent (Type = 'robot';
LifeCycle.Phase.Start=/= LifeCycle.
Phase.Finish)
is an instance of such simple logical statement.
In this statement the semicolon stands for logical conjunction. Different types of logical connec-tives may be used in CONCEPT statements. On the one hand, the knowledge pieces interaction may be treated with the use of generalised families of connectives (see for example [7]) like triangular norms and OWA-operators. An example of a parametered family is a strict t-norm

$$H_\gamma(x,y) = x\,y / \gamma + (1-\gamma)(x+y-x\,y), \quad 0 \leq \gamma < \infty$$

Here different parameter values may be interpreted as the strength of the knowledge pieces interaction. On the other hand, a typical case of the agents knowledge inconsistency results from the agents position asymmetry, when an agent has some priority with respect to another one. In this case the non-monotonic operations over the knowledge pieces are needed. These operations may be seen as extensions of standard connectives, where the priority of an operand (that means generally the lack of commutativity and associativity) is established via some indices of knowledge overlapping or subsumption. The main idea is that by modifying the values of a conditional measure $M(K_i|K_j)$ one may manage the agents knowledge consistency. Here the con-ditional possibility measure $\Pi(K_i|K_j) = \sup \min \{K_i, K_j\}$ or the conditional necessity measure $N(K_i|K_j) = 1-\Pi(\rceil K_i|K_j)$ may be taken. Then, for example, the non-monotonic knowledge conjunc-tion may be defined as

$$K_i \wedge_{nm} K_j = \{(1-\Pi(K_i|K_j) \wedge K_i\} \vee (K_i \wedge K_j)$$

So $\Pi(K_i|K_j)$ is seen as the agents knowledge con-sistency degree, and $1-\Pi(K_i|K_j)$ -as the agents knowledge conflict degree. If $\Pi(K_i|K_j)=1$, then the agents knowledge is completely consistent, and we reduce ourselves to standard conjunction. On the contrary, if $\Pi(K_i|K_j)=0$, $K_i \wedge_{nm} K_j = K_i$, and the knowledge K_j transferred by the agent A_j is in-consistent with the knowledge K_i of the agent A_i.
Furthermore, by using *Believes* attribute we can also express different nested belief statements, such as «Agent *a* believes that before agent *b* believed that agent *c* had done the action of grasping»:
a.Believes = (b.Believes = (c.Action = 'grasp';
c.Por.Time<X); b.Por.Time = X; X<now)).
Here Por is a built-in attribute standing for a point of reference, and X is an existential variable (unknown).
The following expression is an example of a CONCEPT production.
X IN Mobile Robot ==> X IN Agent; X.Life-Cycle.Phase.Start =/= X.LifeCycle.Phase.Finish
This expression is one of four productions asso-ciated with the above mentioned logical state-ment.
Structural statements from a conceptual scheme *S* define heads of relational tables with the attributes occurring in these statements. These tables are filled in so that their rows represent instances and/or counter-instances of *S* concepts. Let us notice that a *state of S* is a set of such tables.
Any production from *S* acts on *S* states; the result of applying the production is a new S state. By using these CONCEPT productions it is natural to formally describe the transition function Φ that

specifies agent's behaviour in multi-agent systems.

3. Some examples of using CBRd

Agents can act by manipulating objects and relations between them. Here the relations are also considered as objects.

Objects are classified by concepts. In CBRd there are two sorts of concepts: classes and binary relations (points of reference are absent). For their specification, there are in CBRd logical sen-tences and default sentences.

Let us consider an illustrative example. Suppose that the agents A_1 and A_2 have the following knowledge concerning animals which are instances of four classes: molluscs, cephalopods, nautili and shell-bearers.

Kno1: Some molluscs are shell-bearers. Cephalopods are molluscs. Some cephalopods are not shell-bearers. Nautili are cephalopods and shell-bearers.

Kno2: Nautili are cephalopods. Nautili as molluscs can be not shell-bearers.

When agent A_2 transfers *Kno2* to A_1, it is necessary to check that *Kno1* subsumes *Kno2*. The subsumption takes place if and only if *Kno1* implies the sentence of «Nautili as cephalopods can be shell-bearers».

Let us write these sentences in the CBRd language:

SOME Mollusc ISA Shell-bearer (1)
Cephalopod ISA Mollusc (2)
SOME Cephalopod ISA –Shell-bearer (3)
Nautilus ISA Cephalopod (4)
Nautilus ISA Shell-bearer (5)
Nautilus AS Mollusc CANBE–Shell-bearer (6)

In the expressions (3) and (6) –Shell-bearer denotes a complementary class; its instances are counter-instances of Shell-bearers.

In a set-theoretic interpretation *I* of the CBRd language, each class name C refers to a subset *I*(C) of the universe *U* and each relation name R refers to a subset *I*(R) of $U \times U$. In particular, let M, S, C, N be abbreviations for *I*(Mollusc), I(Shell-bearer), I(Cephalopod), and I(Nautilus), correspondingly. Then the sentences (1) - (6) are true in the interpretation *I* if and only if $M \cap S \neq \varnothing$, $C \subseteq M$, $M \cap -S \neq \varnothing$, $N \subseteq C$, $N \subseteq S$, ($N \subseteq C$ & $C \cap S \neq \varnothing$), correspondingly.

It is easy to prove that the following rules of inference are logically sound.

$$\frac{\text{X ISA Y, Y ISA Z}}{\text{X ISA Z}} \quad \text{(TR, transitivity rule)}$$

$$\frac{\text{X ISA Y, SOME Y IS Z}}{\text{X AS Y CANBE Z}} \quad \text{(DR1, first default rule)}$$

$$\frac{\text{X ISA Y, Y AS Z CANBE V}}{\text{X AS Z CANBE V}} \quad \text{(DR2, second default rule)}$$

Here is an inference of (6) from *Kno1* = {(1) – (5)}:

1.Nautilus AS Cephalopod CANBE –Shell-bearer
% from (4) and (3) by DR1 %
2. Nautilus AS Mollusc CANBE –Shell-bearer
% from (2) and line 1 by DR2 %
We also infer the sentence
Nautilus AS Mollusc CANBE Shell-bearer (7)
3. Nautilus ISA Mollusc
% from (4) and (2) by TR %
4.Nautilus AS Mollusc CANBE Shell-bearer
% from (1) and line 3 by DR1 %

Clearly, the sentence (7) is weaker than the sentence (3) and then (7) can be deleted. Note that the sentences (6) and (7) are consistent. On the other hand, the sentence

Nautilus ISA -Shell-bearer, (8)

more stronger than (7), cannot be deduced from *Kno1*. Thus, the pair of explicitly inconsistent sentences (5) and (8) never can be obtained.

R e m a r k. This approach to default reasoning is based on original ideas of A.Hautomaki[8]. However, A.Hautomaki considered only two rules of inference: TR and (a variant of) DR1. But it is clear that the set {TR, DR1} of rules is not complete as a deductive system, even for the language of classes with the connectives ISA, SOME-ISA, and AS-CANBE.

Let us consider how to compile CBRd sentences of the simplest form A ISA B. It is clear, if *e* is an instance of the concept A, then *e* must be also an instance of B. Therefore, we have the produc-tion X IN A ==> X IN B (the production scheme, more exactly). On the contrary, if *e* is a counter-instance of B, then *e* must be an counter-instance of A. Therefore, we have the production X IN –B ==> X IN –A.

So, the sentence (2) is compiled into the following two productions:

X IN Cephalopod ==> X IN Mollusc (9)
X IN –Mollusc ==> X IN –Cephalopod (10)
The sentence scheme SOME A ISA B is compiled into the following production scheme:
==> c IN A; c IN B,
where c is new (Skolem) constant, i.e. when compiling just another sentence from an object text (for compiling) the constant c is replaced by the constant c_j where j is the minimal number i such that c_i does not enter into preceeding sen-tences. For example, two SOME-ISA sentences from the object text (1)-(6) are compiled into productions
==> c1 IN Mollusk; c1 IN Shell-bearer (11)
==> c2 IN Cephalopod;c2 IN -Shell-bearer (12)
The sentence scheme A AS B CANBE C is compiled into the following production schemes: ==> c IN B; c IN C,
X IN A ==> X IN B,
X IN –B ==> X IN –A.
The CBRd language also has the connective NOT playing the role of negation of sentences. For example, the sentence NOT A AS B CANBE C is true under an interpretation *I* if and only if it is not true that $I(A)\subseteq I(B)$ and $I(B)\cap I(C)\neq\varnothing$, i.e. either a) not $I(A)\subseteq I(B)$ or b) $I(B)\cap I(C)=\varnothing$.
Let Sign denote the special class such that $I(\text{Sign})\subseteq\{+,-\}$ for any interpretation *I*. Suppose the sign + corresponds to the case (a) when there exists an instance *e* of A such that e is counter-instance of B. Then we have the production sche-me + IN Sign ==> c IN A; c IN –B.
Let us rewrite case (a) as I(B) < I(–C); then we have two production schemes
– IN Sign; X IN B ==> X IN –C,
– IN Sign; X IN C ==> X IN –B.
To prove that (6) follows from Knol={(1)-(5)}, we take the negation of (6),
NOT Nautilus AS Mollusc
CANBE -Shell-bearer, (13)
add it to *Knol* and compile *Knol* ∪ (13). This results in productions (9) - (12) together with the productions listed below.
X IN Nautilus ==> X IN Cephalopod (14)
X IN –Cephalopod ==> X IN –Nautilus (15)
X IN Nautilus ==> X IN Shell-bearer (16)
X IN –Shell-bearer ==> X IN –Nautilus (17)
+ IN Sign ==> c3 IN Nautilus (18)
– IN Sign; X IN Mollusc ==>
X IN Shell-bearer (19)
– IN Sign; X IN –Shell-bearer ==>
X IN –Mollusc.
(20)

So, the result of compiling *Knol* ∪ (13) is the set of productions PS={(9)-(12),(14)-(20)}. Now we can apply PS iteratively, starting with the initial fact base BF(0):
BF(t+1) = result of applying PS to BF(t).
Two steps of the iteration are shown in the Table.

Table

Sign	Mollusc	Cepha-lopod	Nautilus	Shell-Bearer
+/a1 –/a1				
	+c1	+c2	+c3/a1 –c3/a1	+c1 –c2
	+c2 –c2/a2	+c3/a1 –c3/a1	–c2	+c1/a2

The initial fact base BS(0)={+/a1 IN Sign, –/a2 IN Sign} is recorded in the first row of the table. Here a1 and a2 denote two possible alternatives. At the first step productions (11), (12), and (18) were applied. The result of applying (11) and (12) consists of the facts +c1 IN Mollusc, +c1 IN Shell-bearer, +c2 IN Cephalopod, -–c2 IN Shell-Bearer; the application of (18) results in the facts +c3/a1 IN Nautilus and c3 IN Shell-bearer. (These facts are recorded in the second row.)
At the second step, productions (9), (14), (17) were applied. After that, the contrary pair of facts +c3/a1 IN Nautilus, –c3/a1 IN Nautilus has appeared. This means that the alternative a1 takes no place. Hence the alternative a2 is realised and we may delete /a2 from all facts. After deleting, the contrary pair +c2 IN Mollusc and –a2 IN Mollusc. Now it means that the fact base BF(2) is inconsistent.

4. The CBRd language

4.1. Syntax

In CBRd there are four syntactic categories: *C-terms* (they denote classes), *R-terms* (denote binary relations), *P-terms* (denote one-place predicates), and *sentences*.
R-terms:
- every relation name is a R-term

- if T_1 and T_2 are R-terms then $-T_1$, (T_1AND T_2), (T_1OR T_2), INV(T_1), ($T_1.T_2$), and ($T_1..T_2$) are also R-terms.

C-terms:

- ALL and NULL are C-terms
- every class name is a C-term;
- if T_1 and T_2 are C-terms then $-T_1$, (T_1 AND T_2), and (T_1 OR T_2) are also C-terms;
- if T_1 is a C-term and T_2 is a P-term then T_1 THAT T_2 is a C-term
- if T_1 is a C-term and T_2 is a R-term then (T_2 / T_1) and (T_1//T_2) are C-terms.

P-terms:

- if T_1 and T_2 are P-terms then $-T_1$, (T_1 AND T_2), and (T_1 OR T_2) are also P-terms
- if T_1 is a R-term, T_2 is a C-term, c is a constant, and X is a variable then T_1EACH T_2, T_1SOME T_2, T_1c, T_1 X are P-terms.

Sentences:

- if q is a sentence and q does not begin with the connective NOT then NOT q is a sentence;
- if T_1, T_2, and T_3 are C-terms or they are R-terms then EXIST T_1, NULL T_1, T_1 ISA T_2, T_1 == T_2, SOME T_1 ISA T_2, and T_1 AS T_2 CANBE T_3 are sentences;
- if T_1 is a C-term, T_2 is a R-term, c and d are constants then cISAT_1 and cT_2d are sentences
- if T_1, T_2 are C-terms and c is a constant then c AS T_1 CANBE T_2;
- if T_1 is a C-term, T_2 is a P-term, and X is a variable then EACH T_1 T_2, SOME T_1 T_2, EACH X T_1, and SOME X T_2 are sentences.

4.2. Denotative semantics

An interpretation *I* assigns to every class name *C* and relation name *R* some subset *I(C)* of a universe *U* and some subset *I(R)* of the Carthesian product $U \times U$ respectively. Any interpreta-tion *I* can be extended (in a standard manner by Tarski method) on C-terms, R-terms, P-terms, and sentences such that $I(T_1) \subseteq U$, $I(T_2) \subseteq U \times U$, $I(T_3)$: $U \rightarrow \{true, false\}$, and $I(q) \in \{true, false\}$ for any C-term T_1, R-term T_2, P-term T_3, and sentence q.

In particular, we define:

- $I(T_1 \text{OR } T_2) = I(T_1) \cup I(T_2)$;
- $I(\text{INV}(T)) = \{(x,y) \mid (y,x) \in I(T)\}$;
- $I(T_1.T_2) = \{(x,y) \mid (x,z) \in I(T_1)$ and $(z,y) \in I(T_2)$ for some z$\}$
- $I(T\ c)(x)$ = true iff $(x,c) \in I(T)$;
- $I(T_1 \text{ SOME } T_2)(x)$ = true iff $(x,y) \in I(T_1)$ for some z;
- $I(T_1/T_2) = \{x \mid (x,y) \in I(T_2)$ for some $y \in I(T_1)\}$;
- $I(\text{NULL } T)$ = true iff $I(T) = \emptyset$;
- $I(\text{EACH } T_1\ T_2)$ = true iff $I(T_2)(x)$ = true for all $x \in I(T_1)$;
- $I(T_1 \text{ AS } T_2 \text{ CANBE } T_3)$ = true iff $I(T_1) \subseteq I(T_2)$ and $I(T_1) \cap I(T_3) \neq \emptyset$; etc.

By definition, a *conceptual scheme* S (in the CBRd language) is a finite set of CBRd sentences.

5. Compilation of conceptual scheme

In Section 2 we have shown how to compile the sentences of the form A ISA B, SOME A ISA B, and A AS B CANBE C into productions. Let us consider one more sentence form, EACH A B SOME C. In a given interpretation *I* the sentence asserts that for each $x \in I(A)$ there exists $y \in I(C)$ such that $(x,y) \in I(B)$. Therefore a Skolem function *f* exists such that $(x, f(x)) \in I(B)$ and $f(x) \in I(C)$ for any $x \in I(A)$. This means that for any object $e \in U$ either $e \in I(C)$ and $(e, f(e)) \in I(B)$ or $e \notin I(A)$. From here the following schemes of productions are obtained:

X IN A ==> f(X) IN C; (X, f(X)) IN B,
f(X) IN –C ==> X IN –A,
(X, f(X)) IN –B ==> X IN –A.

By the same argumentation line we determine corresponding production schemes for all sentence schemes that are necessary to construct arbitrary CBRd sentences. This correspondence is represented in the table used by the compiler of conceptual schemes.

Let *Pr(S)* denote the production set which results from compiling a conceptual scheme S. Here *Pr(S)* acts on fact bases with the facts of the form +*e* IN C or –*e* IN C where C is a concept name and *e* is a member of the Herbrand universe H(S) associated with S.

The Herbrand universe H(S) is the set of object terms τ and pairs (τ, σ) of object terms; the set of object terms being generated from object names, Skolem constants and functions extracted from the *Pr(S)*. A good few deduction algorithms for conceptual schemes S in CBRd can be designed with basing on various strategies of applying *Pr(S)* to fact bases.

6. Conclusion

The use of a special CONCEPT language to describe intelligent agents is proposed. This language has advanced facilities to describe the agents mental properties due to its concept-orientation and, specifically, through the points of reference and coreferentiality relation.

Moreover, the creation of multi-agent systems supposes the agents collaboration that requires knowledge updating and sharing. So the problem of detecting knowledge subsumptions and inconsistencies arises that demands the development of efficient procedures to maintain knowledge base soundness. This problem is faced in this paper, where a CONCEPT subset - the CBRd language, being an extension of the CBR language - is considered. A method of compiling declarative knowledge into a set of production rules acting on fact bases has been developed. In a sense it is close to a well-known analytical tableaux method.

Bibliography

1. Shoham Y. Agent-oriented programming. Artificial Intel- ligence, vol.60, №1, 1993.
2. van den Akkert J., Siebes A.. DEGAS: a temporal active data model based on object autonomy. Techn. rep.CWI, Computer Sci.,Amsterdam, The Netherland, 1996.
3. Woods W.A., Schmoltze J.G. The KL-ONE family. Computer Math.Applic., vol.23, №2-5, 1992
4. Plesniewicz G.S. A concept-oriented language for applied semiotics. Proc. of IMACS Multiconference CESA' 96, Lille, France, 1996.
5. Плесневич Г.С. Логика моделей «Классы – бинарные отношения». Известия академии наук: Теория и системы управления, I, №5, 1997 и II, №5, 1998.
6. Smullyan R. First-Order Logic, Springer-Verlag, 1966.
7. Нечеткие множества в моделях управления и искусственного интеллекта / А.Н.Аверкин, И.З.Батыршин, В.Б.Тарасов и др. – М.: Наука, 1986.
8. Hautomaki A. A conceptual space approach to semantic networks. In: F.Lehman (ed.) Semantic Networks in Artificial Intelligence, Pergamon Press. 1992.

ABOUT CONVERSATIONS BETWEEN MULTIPLE AGENTS

Pierre–Michel Ricordel, Sylvie Pesty, Yves Demazeau

Leibniz IMAG, 46 avenue Félix Viallet, 38031 Grenoble cedex, France
{Pierre–Michel.Ricordel, Sylvie.Pesty, Yves.Demazeau}@imag.fr

Abstract

Current interaction languages between agents use formally specified semantics and protocols to allow good interoperability and understanding between agents. However, such a rigid interaction model restrains interaction capabilities of agents. In this paper we propose a more cognitive interaction model which allows open discussions between agents without strictly predefined rules, but with a cognitive conversational tracking. We study this interaction model in its theoretical aspects and practical aspects, first in the case of a conversation between two agents, then in the case of a conversation between several agents.

Keywords: *interaction, conversation, dialogism, meaning*

1. Introduction

Nowadays, interactions between agents generally use a formally defined interaction language and interaction protocols. However, such a rigid interaction model raises difficulties, particularly in the case of interoperability of heterogeneous agents or noisy communication channels. In addition, we note that human interactions do not follows such strict rules, but allow good interactions between anybody even in a noisy environment. In this paper, we propose to briefly present a human–inspired interaction model called "dialogism", and our efforts to port this cognitive–science theory to an applicable interaction model for multi–agent systems. Furthermore, we focus on the case where more than two agents interact together.
The choice of a human–inspired interaction model seems relevant since human–inspired metaphors have always been a great source of inspiration in Artificial Intelligence, and since we can experiment in every day life that people can interact among themselves despite the fact that they have different experiences of the language, contrary to agents.
We first briefly present the "dialogism theory" from a cognitive–science point of view, we then discuss the benefits that multi–agent systems can earn of using dialogism.
Afterwards, we present our effort to apply this theory to define an interaction model for multi–agent systems, specifically in the context of a conversation between two agents, first its theoretical aspects, then its practical aspects. We also present experimental results from a dialogic prototype.
Then, before concluding, we propose to extend the model presented in the second part to allow more than two agents to discuss together, presenting results from the adaptation of the previous prototype.

2. Introducing Dialogism

2.1 The origins of dialogism

Very early, linguists, philosophers, semioticians have perceived the fact that the meaning of a message (which can be a speech act, but also a gesture, a work, etc...) is not universal, but is co–constructed by the two interactors. More recently, the semiotician and writer Umberto Eco analyses "*The interpretative co–operation in the narrative texts*" [Eco77]. Dialogism and its ideas have also been studied in linguistics, management [Lei95] and communication, but is a quite new concept in multi–agent systems [CRPD98].
We have based our works on a conversational model conceived at the Laboratory of Interaction Psychology at the university of Nancy 2, which is called *interlocutionary logic* [Bra92, Bra94, TB92]. The interlocutionary logic is a dialogisation of the general formal semantics proposed by Vanderveken [Van 88, Van90, Van91], itself being an enrichment of the illocutionary logic [SV85].
The base of these theories is that any sentence is an act, a speech act, and can be written in a $F(P)$ form, where F is the illocutionary force and P the propositional content. The interlocutionary logic is based on these works, and integrates non literal aspects, in a radically dynamic perspective. This logic is described in [Bra94]. Briefly, this conversation model postulates that there is not a single meaning exchanged by the interactors, but a potential of meaning, and that this potential of meaning is actualised by the two interactors in a

dynamic and collective way, along the conversation [BP99].
However, if interaction languages in multi–agent systems generally integrates elements of this illocutionary logic, it totally ignores the new aspects of the interlocutionary logic. In the next section we present some reasons for using dialogism as an interaction model for multi–agent systems.

2.2 Why dialogism in MAS ?

Classic interaction languages like KQML [FFKE94] or ACL FIPA [FIPA97] are not so far from a Shannon–like communication model, where data (meaning) is encoded by the emitter (the addresser), carried by the medium, and then decoded by the receiver (the addressee), even if they introduce some elements of the speech act theory, formulated by Searle and Vanderveken in [SV85]. The decoding function is exactly the inverse function of the encoding function, otherwise communication is not possible. This approach of the communication encounters its limits in multi–agent systems, where the strict definition of interaction languages is a bridle for the interoperability of open multi–agents systems. Indeed, if we look at the (short) history of standard agent communication languages, we see that KQML, with its informal description of semantics in natural language, suffers from several incompatible flavours emerging from the standard. To solve this problem, the FIPA foundation has specified a communication language called ACL, grounded on formally defined semantics, which avoid any misinterpretation of semantics. However, such strictly defined semantics constrains the agent model too much, and impose a unique rigid model on the agent developer. Furthermore, strict syntax, semantics and protocols can easily lock interactions in a noisy environment, as it can be observed in the domain of robotics.
Moreover, it may seem strange that agents which are said to be *cognitive*, and by definition autonomous and free in their actions, are not able to interpret messages using their own beliefs and knowledge, but have to use instead a quite *reactive* interaction model. The concept of interaction protocols is unsatisfactory for the same reason : interaction protocols should be an emergent property of open interactions between cognitive agents, but not a restriction on the communication capabilities of agents.
The dialogic interaction model we present in this paper, discharges the interactions from semantics and protocol constraints, and compensates this freedom by a dynamic cognitive conversational tracking. Indeed, research on human communication raises the fact that the meaning of a message is not fixed and unique, but depends on the meaning the addresser gives to it, but also (and especially) the meaning the addressee gives to it.
The following section describes our work trying to adapt dialogism to multi–agent systems in the case of a conversation between two agents [Ric98a].

3 Dialogism as an interaction model between two agents

3.1 Theoretical aspects

In this section, we present the two key points that we extract from the dialogism theory : the 'potential of meaning' and the 'conversational tracking'.

The potential of meaning

Meaning is co–constructed by two interactors (considered as being sincere and rational). More precisely, the meaning of a sentence is not *unique*, but *potential*. The addresser expresses a message, that carries a meaning which is not universal, but seems relevant to the addresser. The addressee interprets the message, which means that he gives a meaning to the message, according to its own beliefs (about himself, the addresser, the language, the conversational context, etc.).

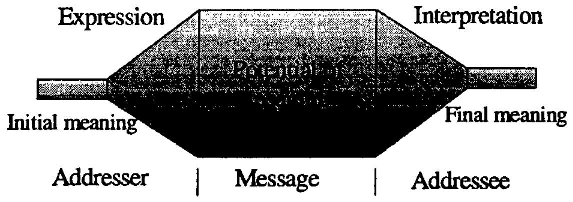


Figure 1 : the potential of meaning

The meaning the addressee interprets is not necessary the meaning the addresser wants to express. However, to allow both agents to understand each other, a second key point of the dialogism is necessary : the *conversational tracking.*

The conversational tracking

An agent has to infer what the agent he was speaking to has interpreted from its message, in order to maintain mutual understanding. More precisely, we can distinguish three different meanings for a single message :
–The meaning the addresser wants to give to its message.
–The meaning the addressee interprets from the message.
–The meaning the addresser thinks the addressee has interpreted. This meaning evolves along the conversation. From the addresser point of view, this meaning is first checked and then replaces the first meaning in the conversation trace if it is validated.
Clearly, the third meaning is the most complex to infer, because it is deduced from the behaviour of

the addressee after he receives the message. But this meaning is a key element for the mutual understanding between the agents. Indeed, to accomplish a successful communication, the addresser has to check if the initial meaning he wants to give to the message and the meaning he thinks the addressee has interpreted are compatible. If these two meanings do not differs too much, the addresser silently solves the conflict, by accepting the new meaning he thinks the addressee has interpreted as the final meaning of the message in its own conversation trace. On the other hand, if these meanings are too different, the addresser explicits the lack of understanding by re–expressing the same meaning in a more precise way, or by expressing explicitly the lack of understanding (As Richard Nixon said : "*I know you believe you understood what you think I said, but I'm not sure you realize what you heard is not what I meant to say...* ").
After presenting the theoretical frame, we present our efforts to implement this dialogism model.

3.2 Tracks of implementation

Assumptions

We observe that dialogism is a highly cognitive interaction model, which requires a high level of intelligence from the agents, especially for the conversational tracking. Conversational tracking requires that the agents own a representation of its interlocutors, a representation of the conversation involving several speaking turns, and a reasoning system on these representations. Hopefully, it is possible to drastically restrain this complexity by a restriction of the ontology of the conversation. With a reduced ontology, the propositional content of a sentence can not be interpreted in a lot of different ways. As an example, in a scheduling ontology, a sentence like "I suggest Monday at 12" is composed of the illocutionary force "suggest" and the propositional content "Monday at 12" [Van90, Van91]. The propositional content can not be misinterpreted, so the dialogic model is too heavy and is useless for such a content. On the other hand, the illocutionary force is still dialogic, and may be interpreted in different ways. But the detection of misinterpretations of the illocutionary force along the conversation is quasi–impossible because the indices of illocutionary force misinterpretation are too small to allow a good conversational tracking.
We choose to use a restricted model of dialogism with no cognitive conversational tracking to allow to study ways to implement the potential of meaning concept in multi–agent systems without having to conceive a complex cognitive agent model.
We also assume in order to simplify the agent's conception, that agents manipulate precise (non dialogic) internal meanings, and that only the inter–agent messages are dialogic.

Operational model

As shown in the previous assumptions, we plan to implement only the potential of meaning concept. We therefore propose a general operational model of this concept, and a way to implement this model with conversation patterns.
The potential of meaning is generated during the *expression* process, which transforms a precise meaning into a potential of meaning, taking into account the agent's knowledge and beliefs about himself, the others, the conversational context and the language itself. The potential of meaning is then transmitted to the receiving agent, who has to reduce this potential to a precise meaning. This is done by the *interpretation* process, which uses the receiver's knowledge and beliefs about himself, the others, the conversation context and the language itself (which are not necessarily the same as the sender's ones) [Ali97, Ric98a]. The precise format of the message is not defined and depends on the implementation chosen for the expression and the interpretation processes. We illustrate how a simple dialogic system should look like in the following figure.

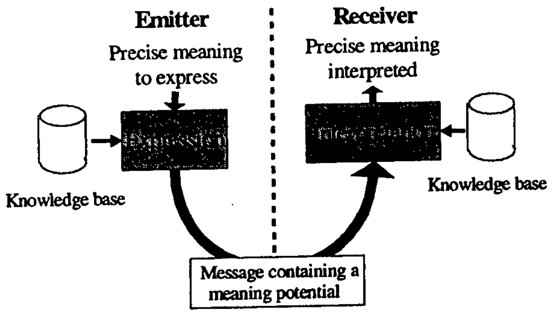

Emitter
Receiver
Precise meaning to express
Precise meaning interpreted
Knowledge base
Knowledge base
Message containing a meaning potential

Figure 2 : The operational model

However, this model does not describe how the expression and interpretation functions have to be implemented. In the following section we propose a way to implement these functions using conversation patterns.

Conversation patterns

The conversion between the precise meaning and the potential of meaning is done by a translation table. The table contains several conversation patterns, which are quite similar to those presented in B. Moulin works [BM97]. A conversation pattern contains a symbol representing a potential of meaning, and the precise meaning attached to it. A conversation pattern can be used in both the expression process and the interpretation process. To express a precise meaning, the agent should choose the conversation pattern which has a precise meaning that matches the best, and then emit the corresponding symbol. To interpret a symbol, the agent should find the corresponding symbol in its conversation pattern and then use the corresponding precise meaning. We can observe that the larger the collection of conversation patterns is, the richer the communication can be. Indeed, a rich collection of conversation patterns allows the agents to exchange subtle meanings. Moreover this collection of conversation patterns has to be private to the agent, that means that to really study the impact of dialogism, the agents should have slightly different conversation patterns. If they are too similar, there is no potential of meaning, if they are too different, communication is impossible. The same phenomenon occurs with human languages : there are common references to a language, but each person has a slightly different perception of the meaning of the words. An example of a conversation pattern is given below. In this example the symbol is "SUGGEST" (we assume that only the illocutionary force is dialogic), and the precise meaning is given by two components of the illocutionary force : the illocutionary point (Assertive) and the degree of strength (0.9).

```
-Symbol : SUGGEST
-Illocutionary point : Assertive
-Degree of strength  : 0.3
```

Note that each agent can have a different precise meaning for the same symbol. A message carrying this potential of meaning can be for instance :

```
<SUGGEST><"Monday 15:30">
```

Where we clearly remark the *F*(*P*) form, with a dialogic illocutionary force associated with a non–dialogic propositional content.

3.3 Experimental results

We have chosen to test this way of implementation in a classical negotiation problem : the automated meeting scheduler [Sen97]. This application domain has the double advantage to be a classical and well documented negotiation problem and to be an usual inter–human discussion subject, which permits us to easily validate the approach by human observations and participation to the dialogues, considering dialogism as a human–inspired interaction model.
We suppose that two agents take an appointment for a day and a time of the week. Each agent has a diary, with an occupation estimation for each hour of the day, between 0 and 1. The goal of each agent is to negotiate an appointment for the lowest occupation time slice. To simplify the analysis of the results, two simple diaries have been used : the "early" diary and the "late" diary. They only focus on one day of the week (monday), the other days are declared "overloaded". The early diary shows a continuous progression of the occupation charge from 8:00 (0.0) to 18:00 (1.0), whereas the late diary shows a continuous decreasing of the occupation charge for the same schedule. The best compromise for both agents is to take an appointment for 13:00.

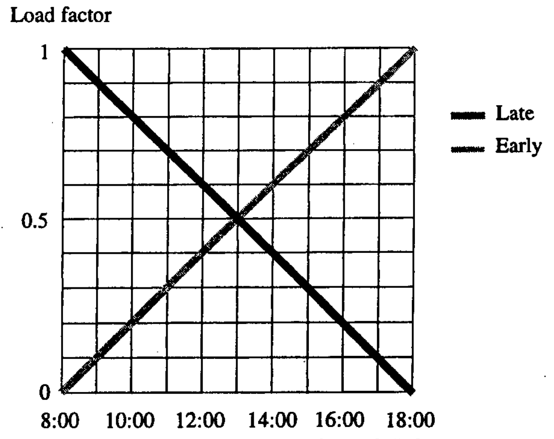


Figure 3 : The "late" and "early" diaries.

The agents use 10 different selected symbols to discuss : *assert, ask, tell, insist, maintain, hypothesise, request, solicit, sustain* and *suggest*, which are chosen from the Vanderveken trees of performatives [Van90, Ali97, Ric98a]. Each agent associates its own meaning to these symbols by the way of their personal conversation patterns. These conversation patterns have been build from a questionnaire filled in by several people. The scheduling part of the agents is constructed using simple scheduling heuristics, which work well enough for this application domain, and allows us to concentrate our efforts on the interaction part of agents. These heuristics are not described here, they simply implement common sense rules of scheduling.
An example of a dialogue between the agent Peter, with the "early" diary, and the agent John, with the "late" diary, is given in figure 4.

John : <Hypothesise><Monday 18:00>
Peter : <Hypothesise><Monday 8:00>
John : <Assert><No>
Peter : <Suggest><Monday 9:00>
John : <Hypothesise><Monday 18:00>
Peter : <Hypothesise><Monday 9:00>
John : <Tell><Monday 17:00>
Peter : <Suggest><Monday 10:00>
John : <Assert><No>
Peter : <Ask><Monday 9:00>
John : <Insist><No>
Peter : <Suggest><Monday 11:00>
John : <Maintain><No>
Peter : <Tell><Monday 12:00>
John : <Suggest><Monday 18:00>
Peter : <Sustain><No>
John : <Hypothesise><Monday 17:00>
Peter : <Ask><Monday 13:00>
John : <Tell><Yes>

Figure 4 : example of dialogue between two agents

The dialogue is rather long because the two diaries do not allow for a trivial appointment, so agents have to converge by "trial and errors" towards the best compromise. Agents evaluate the force of the answers to their propositions in order to guess the other's diary, but each agent has its proper perception of this force, according to its conversation patterns. When an agent accepts a proposed date, the appointment is made, and the dialogue is finished.
We observe that despite the absence of protocols (a question can follow another question), and the shifts of meaning between the two agents, the dialogues always converge near the best compromise.
We have presented dialogism in the case of a discussion between two agents. However, multi-agent systems are generally composed of more than two agents, so we should adapt dialogism to take into account this particularity [Ric98b].

4. From two to several : the multi–dialogism

The dialogism theory is applicable to the dialogue between two agents. In this section, we will study which extensions are needed in order to generalise the dialogism for a dialogue between more than two agents.

4.1 Theoretical aspects

The potential of meaning

The first key element of dialogism, the potential of meaning, is not really affected by multi–dialogism, the only important point is that for multi–dialogism, a single potential of meaning is interpreted in several different ways by different agents. This multiplication of interpretations for the same message is not without influence for the second key point of dialogism, the conversational tracking.

The conversational tracking

Indeed, if we look at the three meanings we have distinguished in section 3.1 for a single message, we can observe that in a dialogue between n agents :
– the meaning the addresser wants to give to its message is already a single meaning,
– there are n–1 meanings that the n–1 addressees interpret from the message,
– the meaning the addresser thinks the addressees have interpreted raises a difficulty.
Indeed, if we suppose that the addresser calculates n–1 of such meanings (one for each addressee), he needs at least n–1 return messages to infer these meanings. Now these return messages are themselves dialogic messages, so each addressee will wait for n–1 return messages to track its meaning (we suppose that every communication is broadcasted). As a result, we may have a combinatorial explosion $((n-1)(n-1)(n-1)...)$ of the number of messages needed to achieve the meaning negotiated and accepted by all the agents. This problem can not appear when only two agents discuss, because when n=2, $((n-1)(n-1)(n-1)...)=1$, reflecting the fact that the meaning can be negotiated among two agents in a finite number of speech turns.
To solve this problem, we need to re–binarise the relations between agents. This binarisation is achieved by the representation that the addresser has of its addressees. Instead of taking in account all its addressees, the addresser takes its addressees as a single audience. This audience is a virtual agent composed of all the addressees, complying with recent research on recursive multi–agent systems. The addresser may use an audience model made out of its knowledge of its addressees, and then infer the third meaning from the feedback of its audience. With this model, dialogism becomes computational again.
If a message coming from the audience reflects a misunderstanding, the addresser can correct it in two different ways, depending on its evaluation of the source of the misunderstanding :
– If the addresser believes that the lack of understanding is due to a too fuzzy expression of the potential of meaning, he will continue to consider its audience as being "unique", and will solve the lack of understanding by re–expressing more clearly the potential of meaning (or at least expliciting the lack of understanding). This strategy may be dangerous if only one addressee of the audience has misunderstood the potential of meaning, because the other addressees who have understood it well may be confused by the re–expression or the explicitation of a lack of understanding.
– If the addresser believes that the lack of understanding is due to the wrong interpretation of a

single addressee in its audience, he can solve the lack of understanding by taking apart the agent which is the source of the lack of understanding, and negotiate the meaning with him by a private communication link. When the lack of understanding is solved with this agent, the dialogue is resumed in broadcast mode. This strategy can be very expensive if several agents in the audience have misunderstood the meaning because the addresser has to re–negotiate the meaning with each of them.
The choice of the good strategy is important, and has to be based on the estimation by the addresser of the origin of the lack of understanding : is it a wrong expression or a wrong interpretation ? The same dilemma occurs when a teacher, seeing that a student misunderstood a lesson, has to decide between re–explaining the lesson to the entire class or re–explaining it to the student in private.
However, these advanced theoretical aspects of the conversational tracking needs to be studied in more detail to issue to a practical implementation.

4.2 Towards implementation of multi dialogism

The levels of communication

We can distinguish three distinct levels of negotiation in dialogue between several entities :
– The speaking turn negotiation : who takes the floor to speak ? This level is necessary because of the asynchronous communication mode induced by the number of agents involved and the protocol–free dialogue context of dialogism.
– The message meaning negotiation : *what does it mean ?* This level is the dialogic meaning negotiation.
– The main negotiation : *what did it propose to me ?* This level expresses the domain dependent negotiation.

The speaking turn negotiation : communication channel and time model

In this context, the word "negotiation" is too strong. Indeed, because speaking turn negotiation occurs at each speaking turn, the negotiation should be extremely simple. In fact it looks more like a simple mutual exclusion algorithm, but behind this algorithm, is defined the environment in which the agents evolves. Before explaining how the agents negotiate the speaking turn, we have to present the communication channel and the time model.
We made the assumption that communication between agents is asynchronous (i.e. each agent can start to speak at any time) and broadcasted. Such communication channel is needed to allow any agent to speak without any global or predefined constraints, like communication protocols. Moreover natural dialogues between humans use such a type of channel, so dialogism, as a human–inspired communication model, has to deal with it.
In this *natural* communication model, time is continuous, and each person may speak at any time. If two persons start to speak (approximately) at the same time, the overlap is detected by the speakers, who then stop speaking and negotiate who speaks first.
This *natural* communication model has inspired a *hardware* communication protocol called Ethernet (IEEE 802.3), in which time is also continuous, and where the computers emit messages on the same wire, which plays the role of the broadcast channel, the overlap of two messages being called a *collision.*
When we try to translate this *hardware* communication protocol into a *software* communication protocol, the main problem we face is that time is usually not continuous in a software environment, but discrete. The time can be continuous in the case of real–time agents, but this hypothesis has to be rejected because it introduces an execution time bias which is undesirable for the needs of the experiment. Thus we need a form of "virtual" time, which is discrete at the computer execution level, and quasi–continuous at the agent level.
This is achieved through the following time model : the time is measured in *time turns.* At each time turn, each agent has in turn the possibility to emit a message to the other agents (in a broadcasted way). Agents are not forced to emit at each time turn. On the contrary, they should not emit too often to avoid collisions. When a message is emitted by an agent, it is retransmitted to every agents at the next time turn, as shown in the figure below.

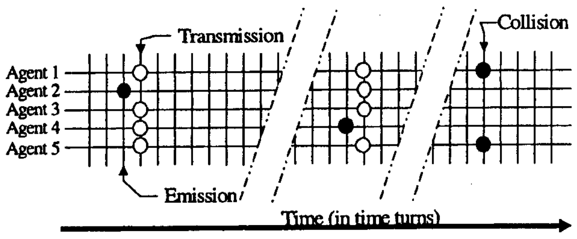


Figure 5 : time model used for the speaking turn negotiation

If two agents emit a message at the same time turn, a collision occurs, and the agents have to re–emit their messages with a random delay, as done in the Ethernet protocol. With a sufficient time–spacing of communicational intentions of the agents, we can diminish statistically the problem of message collisions.
With this model, we reach our goals : all agent has an execution slice (of any real–time duration) at each time turn, but from the point of view of the

agent, the virtual time seems quasi–continuous.

The message meaning negotiation

This negotiation level corresponds to the dialogic meaning negotiation. As we have discussed before, the transition from two to several dialoguing agents entails modifications to the theory mainly on the level of the conversational tracking, but not on the level of the concept of potential of meaning. So, as we assume in our operational model that we implement only the potential of meaning concept, the operational model does not need further modifications in order to work in a multi–dialogic context. Also, the conversation patterns implementation works fine in this context. However, beyond the simple potential of meaning concept, any form of conversational tracking or conversation representation should be modified to take into account the capability of several agents to discuss together..

The main negotiation

This negotiation corresponds to the goal of the conversation between agents, and depends on the chosen application domain. It is up to the multi–agent system's conceiver to define negotiation rules that deal with the fact that several agents participate at the negotiation.

4.3 Experimental results

We have applied the multi–dialogism model to the previously presented scheduling application. The new communication channel and time model have been implemented, and agents have been slightly modified to work with it. Agents listen at the communication channel. When they receive a message, they compute a corresponding answer and its strength. They then wait for a time inversely proportional to this strength before answering. If a message is sent by another agent in this timespan, the agent resigns its scheduled answer and recalculates another one corresponding to the new message. So the dialogue is open, and only the most important messages are expressed (more precisely the messages that their senders think to be important). The "audience" concept presented in section 4.1 is also used. Indeed, agents do not know exactly who participates at the negotiation, and represent other agents as a unique synthetic agent. This allows a fully decentralised negotiation, where agents can enter or leave the discussion at any time. The conclusion of the negotiation is also decentralised. Indeed, when an agent has accepted a proposed date, he will never change its point of view until another agent refuses or proposes another date. So when all agents have accepted a proposed date, none of them have communicational intentions, and the consensus emerges when the communication channel becomes silent for a long period of virtual time.
An example of a dialog between the agent Peter, with the "early" diary, the agent Henry, with the "early" diary, and the agent John ,with the "late" diary, is given in figure 6.

John : <Hypothesise><Monday 18:00>
Peter : <Suggest><Monday 8:00>
John : <Assert><No>
Peter : <Ask><Monday 9:00>
John : <Assert><No>
Henry : <Hypothesise><Monday 10:00>
John : <Assert><No>
Peter : <Tell><Monday 11:00>
Henry : <Sustain><Yes>
John : <Suggest><Monday 18:00>
Peter : <Sustain><No>
John : <Maintain><No>
Henry : <Ask><Monday 12:00>
John : <Insist><No>
Peter : <Tell><Monday 11:00>
Henry : <Sustain><Yes>
John : <Hypothesise><Monday 17:00>
Henry : <Affirm><No>
Peter : <Suggest><Monday 11:00>
John : <Assert><No>
Peter : <Affirm><Monday 13:00>
John : <Insist><Yes>
Henry : <Tell><Yes>

Figure 6 : Example of dialog between three agents.

We observe that in that case, a single acceptation is not enough to conclude the negotiation. All parties have to accept the same date. When it is done, the dialogue stops by itself. We also observe that the dialogue is longest than in a dialog of two agents, because of the number of agents involved, but this lengthening is limited by the speaking turn policy which allows to express the strongest constraints sooner than the weakest ones.

5. Conclusion and perspectives

Dialogism is a promising interaction model, which allows open conversations between agents without any predefined interaction constraints, enabling an easiest interoperability between agents, a lower sensibility to communication noise, and also a more natural interaction between humans and agents. However, the dialogism theory is so complex that only a small subset, the potential of meaning, has been implemented and tested. Conversational tracking, which is the core of dialogism, is still at a theoretical level, and needs to be better formalised. Even if dialogism, as a human–inspired interaction model, seems to require a very high level of intelligence from the agents, the gains it brings to multi–agents interaction are so promising that we can not step aside of it, possibly at the expense of some adaptations to better integrate it into multi-agent systems and to reduce its complexity.

References

[Ali97] Alidra L., 'Les langages d'interaction dans les systèmes multi–agents', DEA of cognitive science report, Institut National Polytechnique de Grenoble, june 1997.

[BM97] Bouzouba K, Moulin B., 'L'implicite dans les communications multi–agents', In J. Quiqueton, M.–C. Thomas and B. Trousse (Eds.), *JFIADSMA'97*, april 1994, Hermès, Paris, 1997.

[BP99] Brassac C., Pesty S., 'Simuler la conversation : un défi pour les systèmes multi–agents', In Moulin B., Delisle S., Chaib–braa B (Eds) ,*Analyse et simulation de conversations : De la théorie des actes de discours aux systèmes multi–agents*, L'interdisciplinaire, Lyon, 1999.

[Bra92] Brassac C., 'Analyse de conversations et théorie des actes de langage', *Cahiers de Linguistique Francaise*, 13, 62–76, 1992.

[Bra94] Brassac C., 'Speech acts and conversational sequencing', *Pragmatics and Cognition*, Vol 2(1), 191–205, 1994.

[CRPD98] Chicoisne G., Ricordel P.–M., Pesty S, Demazeau Y., 'Outils et pistes pour la pratique du dialogisme entre agents', *6th Journées Francophones d'Intelligence Artificielle Distribuee et Systèmes Multi–Agents, JFIADSMA'98*, Pont–à–Mousson, november 1998.

[FFKE94] Finin T., Fritzon R., Mac Kay D. and Mac Entire R., 'KQML as an agent communication language', *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, ACM Press, New York, 1994.

[FIPA97] Fondation for Intelligent Physical Agents, Agent Communication Language, FIPA97 Specification, Part2, Geneva, October 1997.

[Eco77] Eco U., *Lector in fabula : Pragmatic strategy in a metanarrative text*,

[Lei95] Leigh A., 'Spiral Pyramids : a new way of looking at communication', Facilities Management'95, Strathclyde Graduate School of Management, USA, 1995.

[Ric98a] Ricordél P.M., 'Influences mutuelles "Organisation/Interaction", une étude du dialogisme', DEA of informatics report, University of Savoie, Chambery, 1998.

[Ric98b] Ricordel P.M., 'Extention du modèle d'interaction dialogique à *n* agents', Magistere of informatics report, Joseph Fourier University, Grenoble, 1998.

[Sen97] Sen S., 'Developping an automated distributed meeting scheduler', IEEE Expert, p.41, July/August 1997.

[SV85] Searle J.R. and Vanderveken D., *Foundation of illocutionary logic*, CUP, Cambridge, 1985.

[TB92], Trognon A., Brassac C., 'L'enchainement conversationnel', *Cahiers de Linguistique Francaise*, 13, 76–107, 1992.

[Van88] Vanderveken D., *Les actes de discours*, Mardaga, Brussels, 1988.

[Van90] Vanderveken D., *Meaning and speech acts, principles of language use*, (Volume 1), CUP, Cambridge, 1990.

[Van91] Vanderveken D., *Meaning and speech acts, formal semantics of success and satisfaction*, (Volume 2), CUP, Cambridge, 1991.

MULTI-STAGE COOPERATION ALGORITHM AND TOOLS FOR AGENT-BASED PLANNING AND SCHEDULING IN VIRTUAL LEARNING ENVIRONMENT[1]

Leonid B. Sheremetov[a,b], Gustavo Núñez[a]

[a]Centro de Investigación en Computación, Instituto Politécnico Nacional,
Unidad Profesional Adolfo López Mateos, Apdo# 75476, México, D.F., C.P. 07738
Fax: (+52) -5- 5862936
e-mail: sher@cic.ipn.mx
[b]St. Petersburg Institute for Informatics and Automation
of the Russian Academy of Sciences

Abstract

In this paper we describe a problem of generating of study plans in the virtual learning environment, which we have named EVA (states for Virtual Learning Spaces in Spanish). Planning of trajectories and scheduling of learning activities in loosely coupled knowledge domains are performed for each student within a number of temporal and spatial constraints. A model of knowledge for each domain is represented as a type of semantic network, called a concept graph. Cooperative agents responsible for local planning and scheduling are associated with each knowledge domain. A planning procedure over AND-OR graphs is used to generate plan alternatives. A number of heuristics are applied while agent is committed to a particular local plan, which is communicated to the agents controlling related domains. Multistage negotiation algorithm provides means by which an agent can acquire enough knowledge to reason about the impact of his local planes and to achieve globally consistent solution. A facilitator agent coordinates agent's activity. Prototypes of agents have been developed using JDK 1.2 and JATLite packages.

Keywords: *multi-agent systems, planning, negotiation, virtual learning environment.*

1. Introduction

Since their conception more than a quarter of a century ago, knowledge-based learning environments have offered significant potential for fundamentally changing the educational process [2, 16, 25]. The key feature of these systems is the possibility to acquire, represent and use knowledge. This knowledge usually contains a model of problem domain, a model of student's beliefs, and a model of teaching strategies and styles. A problem domain model is an agglutinating center, which relates the concepts to be taught, is used to define student's state in the knowledge space and to find solutions and applicable rules to present knowledgeable feedback to students. Thereby it allows to include learning, reasoning and activities planning capabilities into the characteristics of virtual learning environments.

Nevertheless, despite of many expectations, few learning environments have made the difficult transition from the laboratory to the classroom, so the development of pedagogically sound tools has been the time challenge. The investigation project, which we have named *EVA (Espacios Virtuales de Aprendizaje* in Spanish - *Virtual Learning Spaces),* applies the methodology and tools of Distance Learning and Intelligent Tutoring Systems to obtain a new paradigm of the Configurable Collaborative Learning [18]. This project is dedicated to the research and development of pedagogic models and information technologies that provide spaces of knowledge, collaboration, consulting, and experimentation for supporting the learning activities of teams separated geographically that maintain common conversations, matters, and projects.

Agent technology is the promising way to approach these problems. The notion of agents is the central part of contemporary learning environments, where they act as virtual tutors, virtual students or learning companions, virtual personal assistants that help students to learn, mine information, manage and sched-

[1] The work is supported by CONACyT and IPN, Mexico

ule their learning activities [1, 9, 14, 17]. The use of intelligent agents is supposed to help to make a further step in developing customized learning experiences composed of customized sequences of units of learning material (ULM), each of them located anywhere [11].

The main purpose of our project is to develop models, architectures and multi-agent environment for collaborative learning and experimentation. The focus of this paper is one of these problems: learning activities planning and scheduling. The conceptual architecture of EVA is structured into four essential knowledge elements formed by four information deposits and a set of programs called Virtual Learning Spaces. These spaces are:

- *knowledge* - all the necessary information to learn,
- *collaboration* - real and virtual companions that get together to learn,
- *consultation* - instructors or assessors (also real and virtual), who give the right direction for learning and consult doubts, and
- *experimentation* - the practical work of the students in virtual environment to obtain practical knowledge and abilities.

To navigate these learning spaces, a learner needs his personal routes (*study plans*) suggested in an automatic manner by EVA. So, the purpose on the planning system is to design a particular learning trajectory for each student in the learning spaces and schedule it in time. At the next stage, personalized books, called Multibooks, are armed by concatenating of selected ULM along the learning trajectory for each knowledge domain. In the same way, groups of students with similar interests are arranged.

The problem of planning and scheduling belongs to the area of combinatorial problem solving, difficult to be efficiently solved in a traditional way, including traditional knowledge-based approach developed during last two decades. Now it seems, that multiagent system (MAS) technology is one of the most promising ways to manage this challenge due to a distributed way of tasks solving [5, 10, 15], where agents make their local decisions on plan fragments and negotiate the global decision.

A distributed way of decision making means that each agent must make his local decision having a deficit of information about environment and other agents, resulting in conflicts between the decisions. The common idea in all distributed artificial intelligence (DAI) contributions is that agents use negotiation for conflict resolution and hence the basic idea behind negotiation is *reaching a consensus*. Probably, the most commonly used negotiation protocols for task and resource allocation and coordination among agents is the Contract-Net Protocol (CNP) and auction-based protocol, both developed for centralized way of resource allocation [7, 21, 24]. Recently, these protocols were investigated by a number of authors, where it was shown that they propose an efficient way for self-interested agent behavior coordination [8, 22].

In this paper, we consider the use of multistage negotiations algorithm among agents of MAS that have to solve a complex combinatorial task of planning and scheduling of learning activities that makes it possible to detect and to resolve subgoal interactions and conflicts. It can be considered as an extension of CNP, because multiple contracting mechanism is applied. The most close task statement is one considered in [5], which was applied to traffic planning and control of a complex communications system. In the framework of their model, a cooperation strategy in which agents iteratively exchange tentative and high level partial results of their local subtasks, was generated. This strategy results in solutions, which are incrementally constructed to converge on a set of complete local solutions, which are globally consistent.

A specific feature of the task statement in this paper, in general, is that we aim at solving tasks of planning and scheduling, which complexity is conditioned by real time, temporal, and other constraints imposed on synchronous collaborative learning activities. In our case agents are not self-interested, they cooperate to satisfy a global goal - to generate acceptable study plans.

In the paper, we focus on the problem of planning of collaborative learning activities, negotiation algorithm, domain agent and MAS architectures, and implementation. The rest of the paper is organized as follows. In section 2, we discuss the problem statement of formation of study plans in the virtual learning environment on conceptual level. The problem is specified as a task of dynamic planning and scheduling of learning trajectories in the knowledge domains represented as a type of semantic network, called a concept graphs. In section 3, a model of problem solving based on goals definition within the constraints is considered. In section 4, the main idea of algorithm of multistage negotiation is described. In section 5, domain planning agent's and MAS architectures are considered and implementation details are discussed. Finally, section 6 is devoted to the discussion of the proposed algorithm in the context of different negotiation techniques. In conclusion we present the main results of the paper and outline directions of future work.

2. Problem statement

At the current stage of experiment, the Virtual Learning Spaces are associated with the knowledge taxonomy of Computer Science at the Master of Science level. To represent this taxonomy we have proposed a model based on the hierarchy of knowledge domains and concept graph representation of knowledge.

2.1 Trajectory planning in the knowledge space

Let us consider the problem of planning of student's trajectories over the knowledge space. The model of knowledge is represented in the form of a concept graph G. The concepts at any level of abstraction (courses, ULM or elementary concepts) and relations between them (uni- and bi-directional, in general) correspond to the nodes and arcs of the graph respectively. Each node i has a number of attributes, including its weight (node ratio) that means the importance to achieve a final goal of learning I_i, knowledge constraints (prerequisites) $P_{i,j}$ and estimated time to learn the concept T_i. A precedence relation $i \rightarrow j$ is used to relate concepts that means that concept *'i'* has to be learned directly before the concept *'j'*. Knowledge constraint attribute $P_{k,j}$ captures a prerequisite relation $k \dashrightarrow j$ that means that concept *'k'* is needed to learn the concept *'j'*, but not necessarily precedes it. Each arc *(i, j)* also has its weight (relation ratio) that means the strength of relationship or the necessity of the previous concept for the next one $N_{i,j}$. All the ratios are estimated from "not necessary" to "obligatory" and are represented using 5-valued numeric ranking scale from 0.2 to 1. To capture the fact that it can be necessary to learn two or more concepts to proceed with the next one, AND arcs are used.

Alternative paths mean different options for learning. Alternative routes are analyzed basing on the following decision criterion (to be maximized):

$\sum_{i=1}^{n} I_i * N_{i,j}$, where n is a number of nodes on the route.

The semantics of this criterion is that generally more long paths are preferable to be selected if and only if they are consistent with a temporal constraint:

$\sum_{i=1}^{n} T_i \leq T_c$, where T_c is a time constraint

Figure 1 shows an excerpt from the domain model for the "Distributed Intelligent Systems" (DIS) course, adapted syllabus of which is shown in Table 1. It can be seen that the ULMs of this Multibook use concepts from a number of other Multibooks as prerequisites: Object Oriented Programming (OOP), AI, Mathematical Logic (ML) and Distributed Systems (DS). It should be mentioned that at this figure from all the prerequisites only a fragment of AI course concept model is shown.

While selecting the alternatives, we are usually interested not in one and the only but in a number of them. An E-conformation is introduced, which means the possibility to initially accept first E alternatives.

It is extremely difficult to generate learning trajectories in a general case even at the level of ULM, because of a concept graph dimensions. So, the idea of the proposed model is to capture the natural differences and similarities between the graph fragments, introducing the concept of knowledge domain.

2.2 Model of knowledge domains

A subset G' of graph G can be divided in knowledge domains if and only if the following conditions are fulfilled. There exist a collection $D_0, D_1, \ldots, D_n$, n≥2, of subgraphs G', for which:

- $D_0 \cap D_i = \varnothing$
- The subgraph D_0 has outgoing edges (of precedence) to each subgraph D_i
- The subgraph D_0 has no incoming edges (of precedence) from any subgraph D_i
- In each domain D_i there exist at least one node that has no outgoing edges (of precedence).

The subgraph D_0 is called the "common domain" for the domains $D_1, \ldots, D_n$. Note that the division of the graph G in knowledge domains not necessary is unique. Domains also can have common (shared) nodes or intersections. This definition can be used to obtain decompositions successively from whatever subgraph.

An example of knowledge domain decomposition is shown in fig. 2. The node N_{16} is the intersection of the domains D_1 and D_3 and pertains to the both of them. The common domain for the domains D_1, D_2, D_3 is $D_0 = \{N_1, N_4, N_7\}$. In the subgraph $G' = \{N_5, N_6, N_{12}, N_{13}, N_{14}, N_{15}\}$ of the domain D_2, $D'_0 = \{N_5, N_6, N_{13}\}$ is the common domain for the subdomains $\{N_{12}, N_{15}\}$ y $\{N_{14}\}$.

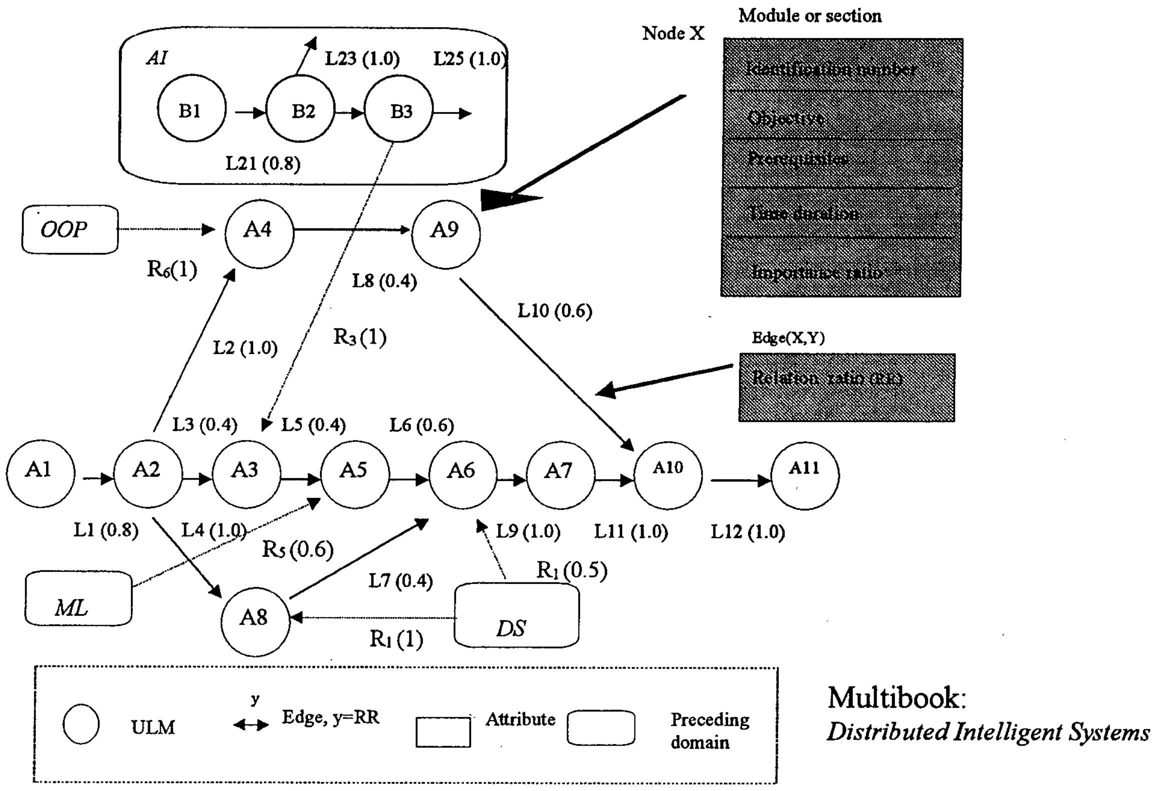


Figure 1. A fragment of a concept graph for a Multibook on DIS

Node number	ULM title	Prerequisites	Importance ratio	Time duration
A1	Introduction to agents and MAS		0.6	4
A2	Agents: definitions y classifications		1.0	4
A3	Shared knowledge and general ontology	Artificial Intel-ligence	0.4	6
A4	Agent oriented software engineering	Object oriented programming	0.8	4
A5	Agent oriented pro-gramming	Mathematical logic	1.0	10
A6	Communication in MAS	Distributed systems	1.0	12
A7	Interaction in MAS		1.0	6
A8	Mobile agents	Distributed systems	0.4	4
A9	Formal specifications of MAS		0.8	6
A10	MAS frameworks and development tools		1.0	6
A11	Examples of software agents		0.6	4

Table 1: A fragment from the "Distributed Intelligent Systems" Multibook syllabus

This model has the following semantics. Knowledge domains have a hierarchic structure that corresponds to different areas and levels of abstraction: common domain, specialty domains, subspecialty domains. For example, common domain consists of OOP, Data Base Design (DBD), Discrete Mathematics (DM) courses, etc. Domain of AI consists of AI, Logic Programming (LP), ML courses, etc., and contains several sub-domains: Knowledge Based Systems (KBS), Automatic Learning, Natural Language, Vision and Robotics. It should be mentioned that the DIS course is an example of the domain intersection, it pertains to the KBS subspecialty, DS and Software Engineering specialty domains.

Usually, courses pertaining to different levels of abstraction can be studied simultaneously. It means that they do not have precedence relations between them. Maximum time duration for each level in the global study plan is also constraint. These temporal constraints are used to schedule student's learning activities.

Initial study plan and learning activities scheduling is generated on the following basis:

(i) Student's initial knowledge state is detected by means of a knowledge prospecting evaluation in the common domain, which is also related to the areas of Computer Science at the graduate level according to the model, proposed by the ACM [4],

(ii) Student's interests in terms of sub-specialty or separated courses from the area, which defines student's final state in the knowledge space.

Initial student's knowledge is considered as initial conditions for the common domain, which means that students even with the same interests have different learning trajectories. The difference between the two types of final state definition is also very important for planning. The later case is more general one, because it can result in plan generation, initiated from different knowledge domains.

Planning process always starts from the final state in a backward chaining manner. Later on, each time a student pass through the exam, the system evaluates his knowledge and tries to infer the reasons of his misunderstandings. It can result in the goal redefinition and, as a consequence, reestablishment of a new study plan, taking his current state into account. Since a student is studying simultaneously a number of courses, the conditions which give rise to goal instantiation may be observed at more than one place on the general concept graph, and the same goal may be instantiated in two or more domains independently.

To perform planning and scheduling within the MAS paradigm, we propose to associate a planning agent with each domain model. These agents must have goal and belief representation capabilities to fulfill the task. Since knowledge domains are interconnected, their local goals and decisions are also interconnected. A multistage negotiation algorithm provides means by which an agent can acquire enough knowledge to reason about the impact of his local decisions and modify its behavior accordingly to construct a globally consistent decision. A special coordinator agent facilitates communication and performs managing activities. This conceptualization can also be expanded to other problem areas, where problem decomposition into domains and subdomains is possible and necessary depending on the problem's complexity.

3. Knowledge representation for planning

In this section we characterize a problem-solving model in which multistage negotiation is useful. For the illustration purposes, throughout the rest of the paper we shall consider a simplified example of study plan generation at the level of ULM, which involves a student, who has to study the "Intelligent Distributed Systems" course. Its concept graph model was discussed in the previous section.

When viewed from a global perspective, plan generation produces a number of alternative study plans for each student. Each plan fragment, as represented in Table 2 for the *DIS* course, is a list of alternating nodes and links, traversing the proposed study paths. Suppose that a planning agent A (responsible for the *KBS* sub-specialty domain) is associated with it. To clarify the example, we have adopted a naming convention for goals and alternative plans, which incorporates the goal and plan number; thus some of alternative plans for the student *alm-1* are designated gl/p1, gl/p2, gl/p3 and gl/p4. The decision criterion value for each plan alternative is shown in the second column of Table 2.

Plan fragments for goal *g1* to plan learning activities for the student *alm-1* (agent A domain):		
Goal/ plan	Decision criterion	Route
gl/pl	5.24	(A1:L1: A2:L4: *'DS'* : A8:L7: A6:L9: A7:L11: A10:L12: A11:L13: A12)

g1/p2	8.2	(A1:L1: [A2:L3: *'AI'* : A3:L5: *'ML'* : A5:L6], [A2:L2: *'OOP'* : A4:L8: A9:L10], [A2:L4: *'DS'* : A8:L7]: *'DS'* : A6:L9: A7:L11: A10:L12: A11:L13: A12)
g1/p3	3.88	(A1:L1: A2:L2: *'OOP'* : A4:L8: A9:L10: A10:L12: A11:L13: A12)
g1/p4	5.24	(A1:L1: A2:L3: *'AI'* : A3:L5: *'ML'* : A5:L6: *'DS'* : A6:L9: A7:L11: A10:L12: A11:L13: A12)

Table 2. Alternative study plan fragments

In this example, two plan alternatives (p1 and p3) have the same value of the decision criterion. For the local conflicts resolution the minimum commitment heuristic rule is applied. It means that the plan alternative with fewer relations with other domains will be given a major priority (p1 in this case).

An agent cannot simply satisfy a local goal by choosing any plan fragment according to the decision criterion, but must coordinate its choice so that it is compatible with those of other agents. Relations with other domains are reflected in Table 2 by including respective prerequisite courses into the learning trajectory notation.

When viewed from the perspective of the system goal, the global study plan appears as an AND-OR tree progressing from the system goal (global study plan at the root), down through goals and plans, to local plan fragments distributed among the agents.

Formulation of a plan as a conjunction of plan fragments induces a set of compatibility constraints on the local choices an agent makes in satisfaction of global goals. An agent generally does not have *complete* knowledge about these compatibility sets. In our application domain, constraints are divided into two categories: local and global constraints. Global constraints are usually temporal ones, e.g. the time at student's disposal to learn the material. Local constraints include a number of students, simultaneously studying the same material, a number of students to form teams (usually from 5 to 7), available resources, etc. A common domain has a special sort of constraints: each plan fragment has to meet student's initial conditions.

From an agent-centered perspective, plan fragment selection is constrained by local resource availability. An agent cannot choose to execute a set of alternative plan fragments that require more local resources than are available. For example, if an agent has already assigned 7 students to go through the second route, it can select only other alternatives because of team size constraint. So, from each agent's perspective, the search is over a group of alternatives subject to a set of local resource constraints and a set of global constraints imposed by actions of other agents.

Multistage negotiation provides a mechanism by which agents coordinate their actions in selecting plans subject to both local and global constraints. Knowledge about domain model, constraint and heuristic rules is stored in the agent's local knowledge base (LKB). Dynamic information, such as plan alternatives and constraints is stored in form of beliefs, which along with capabilities and committed alternatives form agent's mental states. As additional constraints are added to an agent's mental states, its local feasibility tree is augmented to reflect what it has learned. Details on constraint representation and satisfaction can be found in [23]. Domain agent architecture reflecting described model is shown in fig. 3.

4. Algorithm of multistage planning

In this section we discuss the general strategy, then provide more details to the role of negotiation in it. To make these concepts concrete, multistage negotiation is applied to the simplified planning problem, which has been discussed.

4.1 A general planning strategy

Multistage negotiation provides means by which an agent can acquire enough knowledge to reason about the impact of his local decisions and modify its behavior accordingly. The algorithm consists of the following stages of planning and scheduling:

1. construction of the space of alternative plans,
2. plan commitment,
3. scheduling of committed plan alternatives.

The first stage of agent's activity is a phase of study plan generation. At this stage, a coordinator agent initializes a terminal agent (that controls a domain model having terminal nodes for study plan - usually a sub-specialty domain agent) to generate alternatives for study plans for the controlled area. Learning routs are generated based on the information contained in the concept graph. While doing that, each planning agent ascertains what alternatives for partial goal satisfaction are locally possible and assigns a rate for each plan according to the decision criterion.

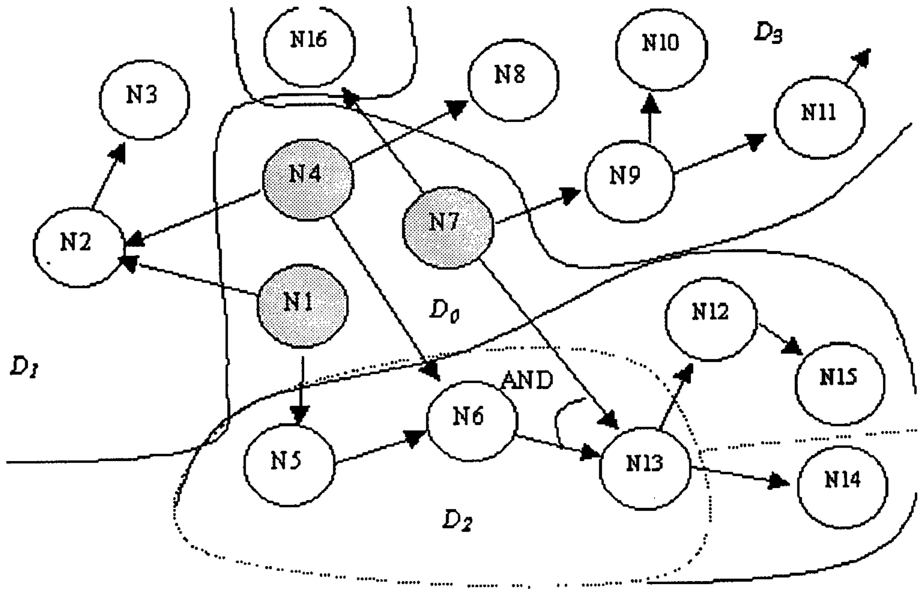


Figure 2. A model of knowledge domains

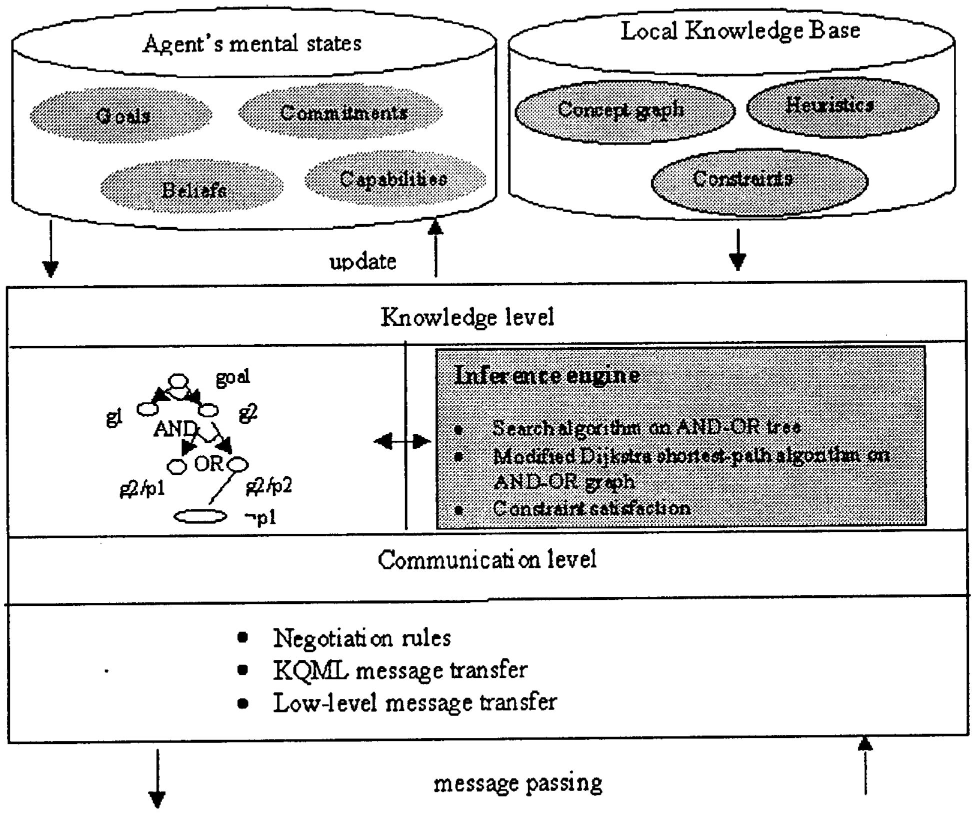


Figure 3. Multilevel architecture of domain planning agent

Which alternative choice obtains the highest rate, depends upon the adopted criterion. In our case, it is the integrated importance of the route with time duration of the route as a cost function. In the case of competitive learning, for example, it can be the most complete study plan (maximum time). Since at this stage, only one stage of negotiation, as in conventional CNP, is used, we shall not consider the details of plan generation here. A modification of a well-known Dijkstra shortest path algorithm is used for each plan fragment generation [6]. These alternative plan fragments are used as a starting point to initialize other agents.

At the next stage, the agent A tenders contracts to appropriate agents for furthering satisfaction of the goals needed to complete established plan fragments. Currently, an active agent provides the coordinator agent with the information on agents from other domains whom to contact. Using E-confirmation principle, the following 3 alternatives will be selected for our example: g1/p1, g1/p2 and g1/p3. According to the applied criterion, the second alternative g1/p2 is the best one. So, as we have mentioned this alternative is related to 3 other domains: AI, DS and common domain. Let us suppose that three other agents, B, C, and D are associated with these domains respectively.

On completion of this phase, a space of alternative plans has been constructed which is distributed among the agents. Coordinator is responsible to establish communications, transmit partial goals and recollect the alternatives, performing facilitator functions. It should be mentioned that a coordinator agent constructs global plans, which have been generated in a distributed manner, and no single planning agent necessarily knows of all plans or any one complete plan. It is also responsible for global temporal constraint satisfaction and partial constraint relaxation.

At the next stage of plans confirmation, each terminal agent examines the goals it instantiated and makes a tentative commitment to the highest rated feasible set of plan fragments relative to these goals. It subsequently issues requests for confirmation of that commitment to agents who hold the contracts for completion of these plan fragments. A protocol applied at this stage can be considered as a multiple CNP. In general case, e.g. when planning occurs for the separate courses, belonging to different domains, each agent may receive two kinds of communications from other agents: 1) requests for confirmation of other agents' tentative commitments, and 2) responses concerning the impact of its own proposed commitments on others. Impact of local actions is reported as confirmation that a tentative local choice is a good one or as negative information reflecting nonlocal resource conflict. The agent rerates its own local goals using the new knowledge and possibly retracts its tentative resource commitment in order to make a more informed choice. This process of information exchange continues until a consistent set of choices can be confirmed. Synchronized global termination is required in our application, so coordinator agent, which receives all the information of the confirmed plans issues a *stop-negotiation* message to all involved agents.

4.2 Negotiation algorithm

When a planning agent begins its activity, it has knowledge of a set of top level goals, which have been locally instantiated. A space of plans to satisfy each of these goals is formulated during plan generation without regard for any subgoal interaction problems. In a general case of planning for a number of students simultaneously, after the phase of plan generation, each agent is aware of two kinds of goals: primary *goals* (or p-goals) and *secondary goals* (or s-goals). In our application, p-goals are those instantiated locally by an agent in response to a query from the coordinator agent for courses from the domain, for which the agent has primary responsibility (because the student is interested in the agent's subregion). These are of enhanced importance to this agent because they relate to system goals, which must be satisfied by this particular agent, if they are to be satisfied at all. An agent's s-goals are those which have been instantiated as a result of a contract with some other agent.

A plan commitment phase involving multistage negotiation is initiated next. A fragment from the agent interaction diagram, illustrating our example is shown in Figure 4. As this phase begins, each node has knowledge about all of the p-goals and s-goals it has instantiated. Relative to each of its goals, it knows a number of alternatives for goal satisfaction. An alternative is comprised of a local plan fragment, points of interaction with other agents (relative to that plan fragment), and a measure of the cost of the alternative (to be used in making heuristic decisions). Negotiation leading to a commitment proceeds along the following lines.

- Each agent examines its *own* p-goals, making a tentative commitment to the highest rated set of locally feasible plan fragments for p-goals (s-goals are not considered at this point because some other agent has corresponding p-goals). g1/p2 is selected in our example.

- Each agent requests that other agents attempt to confirm a plan choice consistent with its commitment. Note that an agent need only communicate with agents who can provide input relevant to this tentative commitment. Agent A communicates with agents B, C and D agents for the p2 alternative.
- An agent examines its incoming message queue for communications from other agents. Requests for confirmation of other agents' tentative commitments are handled by adding the relevant s-goals to a set of active goals. Responses to this agent's own requests are incorporated in the local belief set and used as additional knowledge in making revisions to its tentative commitment.
- The set of active goals consists of all the local p-goals together with those s-goals that have been added (in step 3). The agent rates the alternatives associated with active goals based on their cost, any confirming evidence that the alternative is a good choice, any negative evidence in the form of nonlocal conflict information, and the importance of the goal. A revised tentative commitment is made to a highest rated set of locally consistent alternatives for active goals. In general, this may involve decisions to *add* plan fragments to the tentative commitment and to *delete* plan fragments from the old tentative commitment. For example, p2 alternative is deleted from the goal set because of a conflict with the goal g2 established by the agent C. Messages reflecting any changes in the tentative commitment and perceived conflicts with that commitment are transmitted to the appropriate agents.
- The incoming message queue is examined again and activity proceeds as described above (from step 3). The process of aggregating knowledge about nonlocal conflicts continues until a node is aware of all conflicts in which its plan fragments are a contributing factor.

Negotiation activity in an agent terminates by a coordinator agent, when he recollect the information on all confirmed or rejected alternatives. Actually, when a planning agent either has no pending activity and no incoming communications or if an attempt is made to return to a previous commitment with no new knowledge from other agents, he advertises a Coordinator about the termination of his activities.

The other issue of importance at this point is related to the quality of the result obtained through negotiation. In the initial negotiation stage, each agent examines only its p-goals and makes a tentative commitment to a locally feasible set of plan fragments in partial satisfaction of those goals. Since each agent is considering just its p-goals at this stage, the only reason for an agent's electing not to attempt satisfaction of some top level goal is that two or more of these goals are locally known to be infeasible, which means that the problem is initially overconstrained.

In subsequent stages of negotiation, both p-goals and relevant s-goals are considered in making new tentative commitments. If the system goal of satisfying all of the p-goals instantiated by agents is feasible, no agent will ever be forced to forego satisfaction of one of its p-goals (because no agent will ever learn that its p-goal precludes others), and a desired solution will be found. If, on the other hand, the problem is overconstrained, some set of p-goals cannot be satisfied and the system tries to satisfy as many as it can. While there is no guarantee of optimality, the heuristics employed should ensure that a reasonably thorough search is made.

5. Multi-agent system architecture and implementation details

Learning activities planning system with multistage negotiation is implemented using JATLite package [12]. It is composed of *n* planning agents and a *coordinator agent,* which inherit their methods from the RouterClient class. Agents are implemented as JAVA applets, so they also inherit methods from the Applet and Frame (to support graphic interface) classes of JDK 1.2 package. For communication, agents use the general set of JATLite KQML performatives, namely, *advertise, tell, reply,* and *ask-if.* At the first stage of experiments, this set was extended by the following performatives, which were introduced for implementation purposes:

- *request-planning,* initial external query to start planning process;
- *start-negotiation,* a broadcast performative announcing the start of negotiation process;
- *stop-negotiation,* a broadcast performative announcing the termination of negotiation process.

At the current stage of experiments this additional set of performatives is also changed to the standard one, e.g. *start-* and *stop-negotiation* functions are transmitted by a *broadcast* performative. The role of coordinator is also to supply the agents with initial beliefs that include information about general constraints and resources, retrieved from the global knowledge base and names of agents responsible for each domain. Another function of the coordinator is to generate global study plans and schedules from the proposals submitted by each agent.

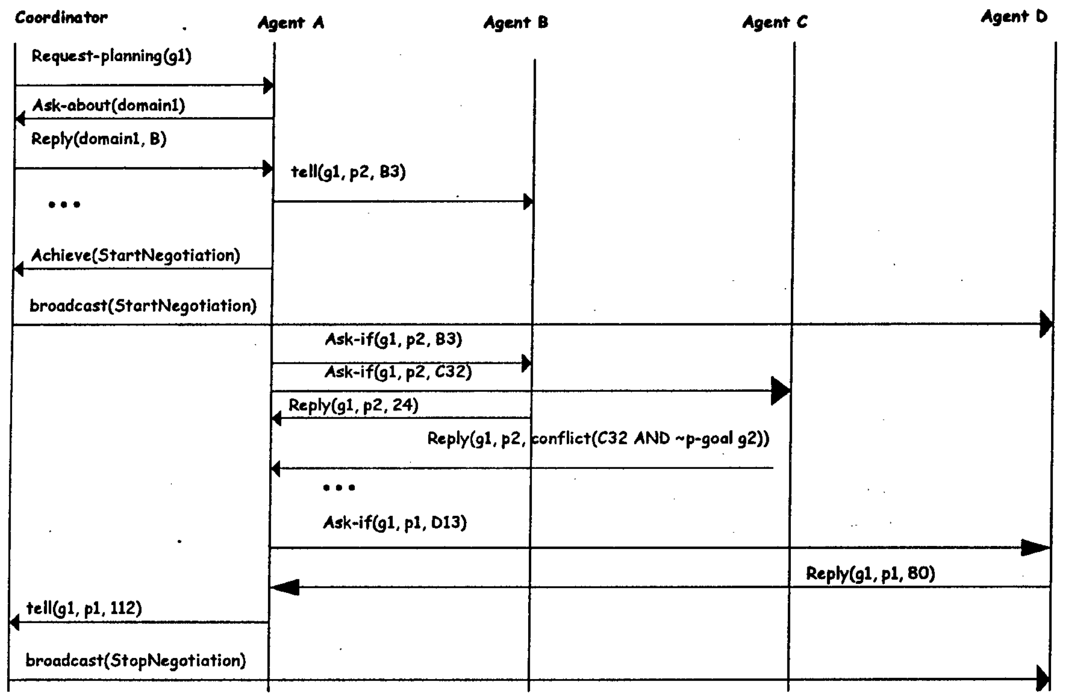
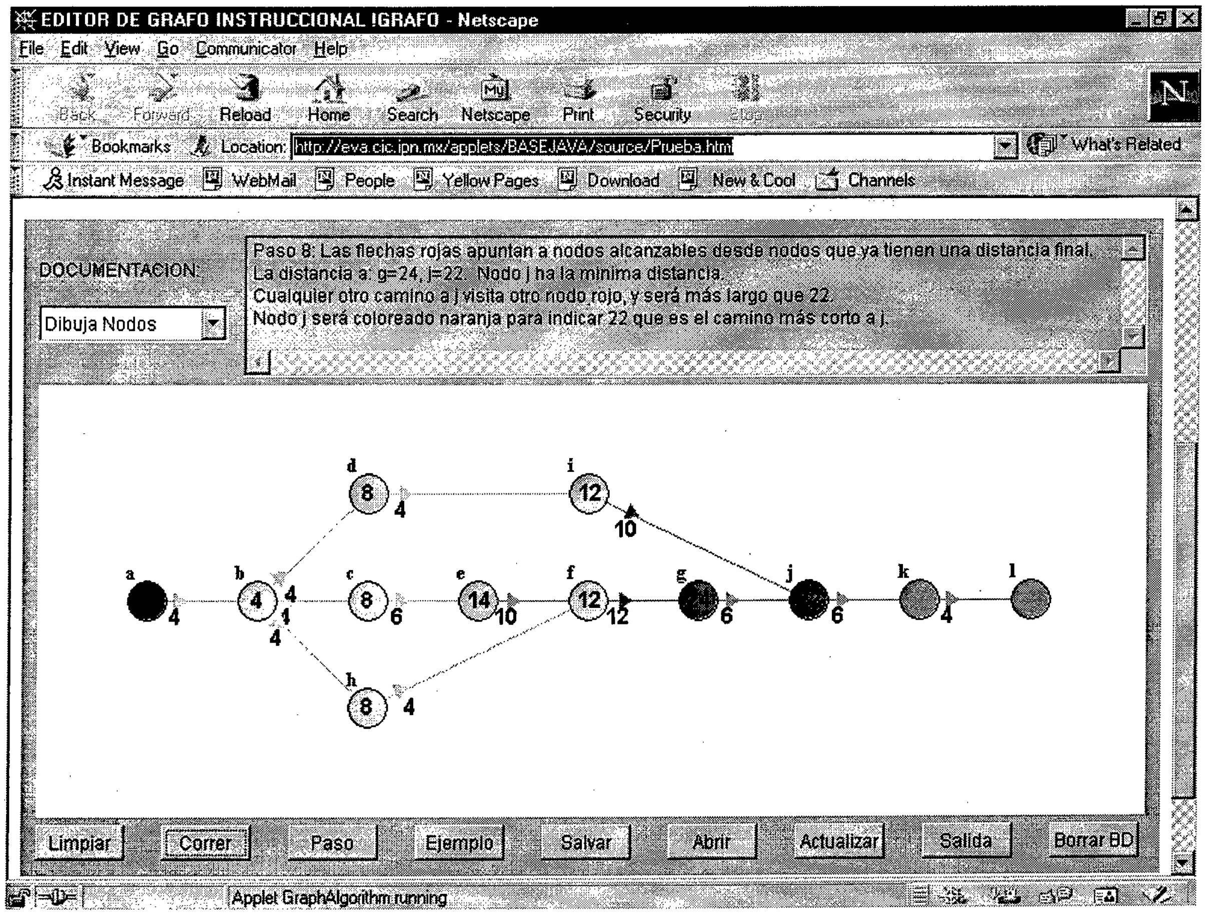


Figure 4. A fragment from the agent conversation diagram


EDITOR DE GRAFO INSTRUCCIONAL IGRAFO - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://eva.cic.ipn.mx/applets/BASEJAVA/source/Prueba.htm What's Related
Instant Message WebMail People Yellow Pages Download New & Cool Channels
DOCUMENTACION:
Dibuja Nodos
d i a b c e f g j k l h
8 4 12 10 4 4 4 4 4 8 6 14 10 12 12 6 6 4 8 4
Limpiar Correr Paso Ejemplo Salvar Abrir Actualizar Salida Borrar BD
Applet GraphAlgorithm running

Paso 8: Las flechas rojas apuntan a nodos alcanzables desde nodos que ya tienen una distancia final.
La distancia a: g=24, j=22. Nodo j ha la minima distancia.
Cualquier otro camino a j visita otro nodo rojo, y será más largo que 22.
Nodo j será coloreado naranja para indicar 22 que es el camino más corto a j.

Figure 5. A screenshot of the user interface of the concept graph development tool.

Figure 5 shows a screenshot of the user interface of the concept graph development tool. Figure 6 shows a screenshot of a planning system, including JATLite Client Applet, Coordinator agent and 4 planning agents (*A-D*) for the situation discussed earlier in this paper. The following four agents are involved in the planning process: *A*, controlling the KBS sub-domain, *B*, controlling AI domain, *C*, controlling DS domain, and *D*, controlling common knowledge domain. For the illustrative purposes, all agents run on the same platform, though really each planning agent runs on the platform, containing domain model files and tools for model development. Agents Graphic Interface consists of two boxes: the left one - for sent messages and auxiliary information (such as committed or conflicting requests), and the right one - for received messages. All the messages maintain KQML format. Agent windows show different stages of negotiation. For example, coordinator agent has already started the negotiation process, having knowledge about existing knowledge route alternatives for each of two sub-goals, established by agents A and C. .

6. Discussion

In this section we shall compare our algorithm with different types and strategies of negotiation. Two types of negotiation is usually considered in the MAS research: Competitive and Cooperative Negotiation, also sometimes associated with decentralized and centralized planning [17]. In the first case, negotiation is used in situations where "agents of disparate interests attempt to make a group choice over well-defined alternatives" [20]. Therefore, competitive negotiation involves independent agents with independent goals that interact with each other. They are not a priori cooperative, share information or willing to back down for the greater good, namely they are competitive. Another type of MAS, includes systems where agents have "a global goal/single task envisioned for the system" [24], in that sense these agents are called "collaborative" [3].

In the conventional CNP approach, a decentralized market structure is assumed and agents can take on two roles, a manager and contractor. The basic premise of this form of coordination is that, if an agent cannot solve an assigned problem using local resources/expertise, it will decompose the problem into subproblems and try to find other willing agents with the necessary resources/expertise to solve these subproblems. The problem of assigning the subproblems is solved by a contracting mechanism. It consists of contract announcement by the manager agent, submission of bids by contracting agents in response to the announcement, and the evaluation of the submitted bids by the contractor, which leads to awarding a subproblem contract to the contractor(s) with the most appropriate bid(s). Although CNP is considered by Smith and Davis as well as many DAI researchers to be a negotiation principle, other researchers believe it is more a standardized coordination method for the following reasons [13]:

1. There must not be a conflict at all between the agents to start the CNP, hence there is no possibility of bargaining between the agents. The manager does not communicate its minimal condition, nor do the bidders have a second choice (no constraint relaxation),
2. A mutual decision is eventually given by the decision of the offer, hence there is no two-way agreement.

To follow this, its limitations involve the fact that it does not detect or resolve conflicts, the agents in the contract net are considered benevolent and non-antagonistic (which in real world scenarios is not realistic).

The last observation seems doubtful because negotiation metaphors can be considered in the both different senses in respect to application. For example, in the well-known auction-based model of agent behavior coordination, agent contractors can be representatives of really competing companies each is self-interested and aimed at each own benefit only. In this case, auction is a mathematical model of a real competition of companies. On the other hand, if we have to solve a centralized planning and scheduling task for activity of a large company the auction-based model is used only as a metaphor that plays the role of a coordination mechanism for partial solutions made in distributed way. For such a case, agent-contractors are not self-interested and aim at solving an only complex task jointly.

Close task statements to that considered in this paper, but for the case of competitive agents in collaborative planning for Air Traffic Control was proposed in [3, 19]. A cooperative negotiation model, in which the participating agents (pilots, air traffic control) collaborate on developing the best plan in terms of the interests of the agents as a group was developed. This model involves an agent detecting conflicts regarding proposed actions and beliefs from other agents and initiating collaborative negotiation to resolve such conflicts. During negotiation, an agent modifies the proposal with appropriate justification for this modification, based on its own beliefs. In this way, agents collaborate to achieve mutual beliefs, thereby resolving conflicts.

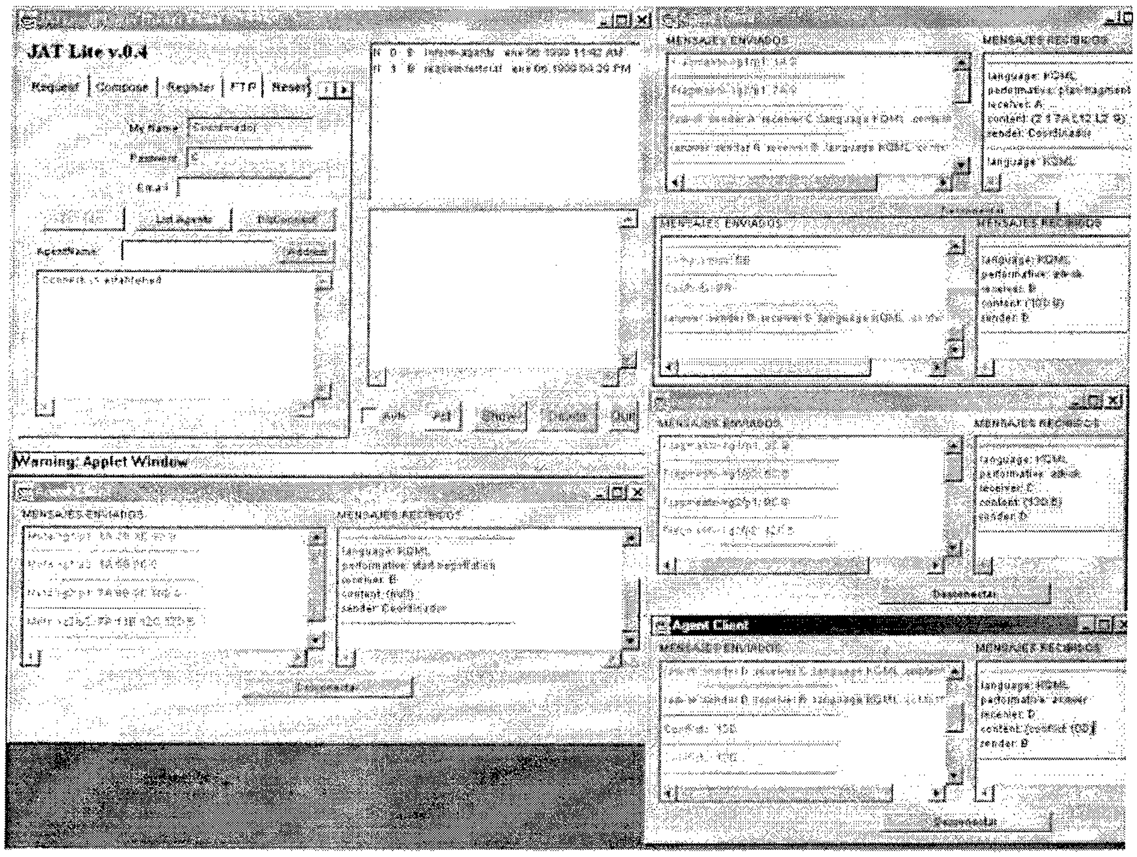


Figure 6. A screenshot of the system with 4 planning agents.

The algorithm discussed in this paper can be considered as a combination of the centralized and decentralized modes of coordination, which are applied at different phases of planning process. While being applied to the case of collaborative agents, it extends the CNP in the following: propose the mechanism for conflict detection, supports iterative exchange of knowledge, and applies constraint relaxation making possible to achieve two-way agreements. That is why we consider it to be a negotiation algorithm, which can be used for coordination purposes of cooperating agents. It also should be noted that this metaphor could be adapted for use in MAS, where agents are self-interested.

4. Conclusion

In this paper, we have presented an approach making use of multistage negotiation algorithm for cooperation in distributed planning and scheduling of students learning activities. It is based on the extension of CNP and permits an agent to acquire enough knowledge to detect conflicts of his local decisions with the others and to negotiate acceptable global decision. The motivation for the development of this cooperation paradigm has the following reasons: (i) subgoal interaction problems that arise in the context of a distributed planning system, (ii) overconstraintness of planning problems, which needs a strategy of constraint relaxation from the global point of view. The last observation seams to be very important, because in many planning problems, the constraints arising from resource availability determines a satisfactory solution to the planning problem. Temporal and resource availability constraints play a crucial role in our system as well.

The distributed planning system discussed in this paper is currently implemented as a full-scale prototype. The protocol of multistage negotiation is under investigation. Planning agents from the student's point of view serve as his personal assistants that assist him to plan and schedule his learning activities. They form a part of the multi-agent environment for individual and collaborative learning with artificial learning companions, personal learning assistants with information filtering capabilities, and agents supporting experimentation activities, which is under development. Cooperation of planning

MAS with personal learning assistants is under consideration. We are also investigating computational efficiency of the proposed algorithm for different planning tasks.

Bibliography

1. Barros Costa, E., Perkusich, A. Modeling the Cooperative Interaction in a Teaching/Learning Situation. *Intelligent Tutoring Systems,* 1996: 168-176
2. Chan, T.W. Learning companion Systems, Social Learning Systems, and Intelligent Virtual Classroom, *In Proc. of the World Congress on AI in Education,* 1996, Vol. 7, No. 2, 125-159.
8. Fisher, K., Muller, J., Heimig, I., & Scheer, A-W., Intelligent Agents in Virtual Enterprises. *In Proc. of the First Intern. Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology,* London, 1996, pp.205-224.
9. Gordon, A. and Hall, L. Collaboration with Agents in a Virtual World. *In Proc. of the Workshop on Current Trends and Artificial Intelligence in Education, 4 World Congress on Expert Systems,* Mexico, 1998.
10. Gorodetski, V. 1997, Basic Ideas of Agent Behaviour Coordination and Auction - based Scheduling Model.. *In Proc. of the I Int. Workshop Distributed Artificial Intelligence and Multi-agent Systems,* St. Petersburg, pp.282-291.
11. Harris, D.A., Online Distance Education in the United States, *IEEE Communications Magazine,* March, 1999, pp. 87-91.
12. JATLite Beta Complete Documentation, Stanford University, 1998, http://java.stanford.edu
13. Jennings, N.R., Paratin, P., & Jonson, M., Using Intelligent Agents to Manage Business Processes. *In Proc. of the First International Conference and Exhibition The Practical Application of Intelligent Agents and Multi-Agent Technology,* London, UK, 1996, pp.345- 376.
14. Kayama, M. and Okamoto, T. A., Mechanism for Knowledge-Navigation in Hyperspace with Neural Networks to Support Exploring activities. *In Proc. of the Workshop on Current Trends and Artificial Intelligence in Education, 4 World Congress on Expert Systems,* Mexico, 1998.
15. Liu J.-Sh. & Sycara, K. Multiagent Coordination in Tightly Coupled Task Scheduling, In *Proc. of the First Int. Conf. On Multiagent Systems,* 1996, pp. 181-188.
16. Mark, M.A., and Greer, J.E. The VCR tutor: Effective Instruction for Device Operation. *Journal of the Learning Sciences,* 1995, 4(2):209-246.
3. Chu-Carroll, J. & Carberry, S., Conflict Detection and Resolution in Collaborative Planning, Intelligent Agents II, Lecture Notes in Artificial Intelligent 1037, Springer Verlag, 1995.
4. Computing as a Discipline, ACM report, 1991.
5. Conry, S. E., Meyer, R. A., & Lesser, V. R. Multistage Negotiation in Distributed Planning, In Bond, A. H. and Gasser, L., (eds.) *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers: San Mateo, CA, 1988, pp. 367-384.
6. Cormen, T., Leiserson Ch., & Rivest R., *Introduction to Algorithms,* McGraw Hill, 1990.
7. Davis, R & Smith, R. G. Negotiation as a Metaphor for Distributed Problem Solving, *Artificial Intelligence,* vol. 20, no. 1, 1983, pp. 63-109.
17. Müller, H. J., Negotiation Principles, In *O'Hare G.M.P., & Jennings, N.R. (Eds.) Foundations of Distributed Artificial Intelligence,* J Wiley & Sons, 1996, pp.211-230.
18. Nuñez, G., Sheremetov, L., Martínez, J., Guzmán, A., & Albornoz, A. The Eva Teleteaching Project - the Concept and the First Experience in the Development of Virtual Learning Spaces. *In Gordon Davies (ed.) Teleteaching'98 Distance Learning, Training and Education: Proceedings of the 15th IFIP World Computer Congress,* Vienna and Budapest, 1998, P. II, pp. 769-778.
19. Rao, A. & Georgeff, M., BDI Agents: From Theory to Practice, In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95),* San Francisco, USA, 1995.
20. Rosenchein, J. and Zlotkin, G., *Rules of Encounter Designing Contentions for Automated Negotiation among Computers,* MIT Press, 1994.
21. Sandholm, T.W., An Implementation of Contract Net Protocol Based on Marginal Cost Calculations, *In Proceedings of 11th AAAI,* 1993, pp.256-262.
22. Sandholm, T.W., Negotiation among Self-interested Computationally Limited Agents. Ph.D. Thesis, 1996. http:// www.cs.wustl.edu/~sandholm/dissertation.
23. Smirnov, A.V. and Sheremetov, L.B. Complex Systems Configuring Based on Multi-agent Technology. *Automatic Control and Computer Sciences,* Allerton Press, N.Y. 1998.
24. Smith R. G., The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, *IEEE Transactions on Computers* C-29(12), 1980, pp. 1104-1113.
25. Youngblut, Ch. Government - Sponsored Research and Development Efforts in the Area of Intelligent Tutoring Systems. Institute for Defense Analyses, USA, 1994.

CONSTRAINT-BASED MULTI-AGENT TECHNOLOGY AND ITS APPLICATION TO THE SOCIAL-ECONOMIC MODELING[1]

Igor E. Shvetsov[2], Tatyana V. Nesterenko[3], Sergey A. Starovit[4], Sergey V. Preis[5]

Russian Research Institute of Artificial Intelligence
& Institute of Informatics Systems, Siberian Branch of Russian Academy of Sciences
Pr. Lavrent'eva 6, Novosibirsk, 630090, Russia
e-mail: [2]shvetsov@iis.nsk.su, [3]nest@iis.nsk.su, [4]serg@iis.nsk.su, [5]serge@iis.nsk.su

Abstract

In the paper, we present a technology for development of dynamic multi-agent systems which is based on a combination of the object-oriented approach with constraint programming. We introduce the notion of an active object as a way of describing and implementing of intellectual, reactive and communicative properties of agents. A special emphasis is made on the dynamic component of this technology. Dynamic systems are classified with respect to their complexity level. It is shown how such systems can be implemented with the help of the technology of active objects. We also present an application of the TAO language to the development of MAS in social-economic modeling.

Keywords:

1. Introduction

The development of multi-agent systems (MAS) is one of the main trends in modern programming (see [1]). We can distinguish two aspects of MAS, technological and conceptual. Technologically, an MAS is constructed as a distributed system of programs that consists of autonomous modules called *agents*. These agents are conceptually characterized by independent behavior as well as by certain intellectual and communicative abilities.

The following three types of agent behavior are usually recognized: reactive, deliberative and cooperative. The first type is characteristic for the most simple agents, which react to events without a logical analysis of the situation and possible consequences. These agents are considered, for example, in the «swarm intelligence» approach [2], where MAS are compared with colonies of insects. The principal idea of this approach stipulates that unconscious joint activities of primitive organisms can support the solution of rather complex, «intellectual» problems.

Deliberative behavior is characteristic for intelligent agents which have certain knowledge of the outside world and can analyze events and predict their consequences using, e.g., logical inference. The behavior of intelligent agents is usually goal-oriented, i.e., they can independently set certain objectives, plan their realization, and be insistent in the implementation of these plans. The most common approach to the specification of deliberative behavior of agents is BDI [3]. It can integrate the potential of all methods of knowledge processing that have been developed over the time of AI existence. However, this approach in a complete form is too complex for implementation or application. Therefore, practical use of BDI often uses an abridged version that has been adapted for a specific subject domain and a class of problems to be solved.

Cooperative behavior of an agent is its ability to interact with other agents in order to solve a common task. As a rule, this interaction consists of negotiations as a result of which agents can enter into contract relationships, form coalitions that can be more or less stable, or simply agree the plans of their actions for a certain period of time. One of the most common negotiation schemes is the auction model. This model is used, for example, in the system MACIV [4], which realizes the selection of

[1] This paper is supported by Russian Foundation for Basic Research (98-01-00694)

companies with various specialties to perform a single construction contract.

There are comprehensive technologies of MAS development which support all three agent behavior types. A notable representative of this class of systems is InteRRaP [5], which is based on the idea of assigning a separate control layer to each behavior type (the so-called layered approach). This approach not only ensures a better organization of the MAS; it also leads to a more efficient implementation. Some disadvantages of the system are the heterogeneity of the tools used to specify various components of MAS, as well as a somewhat low level of some of these tools.

The objective of our work was to develop a high-level, comprehensive technology of MAS development that is based on one of the most advanced branches of constraint programming, called subdefinite calculations (or SD-method, for short). This method was proposed in the early eighties [6] and has been successfully applied in diverse application areas, such as scheduling [7], knowledge processing [8], CAD/CAM and engineering [9], solution of complex mathematical problems [10], and others. Our objective is to extend the SD-method to the class of multi-agent systems.

The technology we have been developing has been named TAO (technology of active objects), since its first implementation [11-13] relied essentially on the ideas of object-oriented programming. This work resulted in a constraint-based language that allowed one to develop mostly multi-agent systems containing reactive agents.

In this paper we present a new version of the language that extends the capabilities for specification of deliberative and cooperative behavior of agents. In addition, the language supports hierarchies of agents and dynamic reconfiguration of MAS. We also present an application of the TAO language to the development of MAS in social-economic modeling.

2. Principal properties of TAO

TAO is a multi-agent technology completely based on the constraint-programming paradigm, which is one of the most promising approaches in the field of artificial intelligence. Constraint programming provides the highest level of problem specification. Instead of algorithms or logical inference rules, this approach uses the notion of a model, which is a set of constraints (relations, equations, inequalities, logical statements etc.) linking the parameters of the problem.

In TAO a multi-agent system (MAS) is created as a collection of active objects (agents) with the following properties:

- The behavior of any active object is described by a set of constraints (i.e., by a model) which defines a space of correct states of the object. An active object is able to choose the optimal solution from the space by analyzing the stream of events coming from outside.
- TAO ensures a high-level specification of the complex models of real-life entities. The interaction between active objects is organized as an asynchronous event-driven process. At each moment an active object can independently change its own state reacting to the events generated by other active objects.
- An abstract global clock is the main active object of any multi-agent system constructed by TAO. It generates the next time moment at which all other active objects can change their states. An active object computes its current state knowing the past (previous states of the MAS) and planning the future.
- Dynamic reconfiguration of the multi-agent system is supported. The number of active objects as well as their links and models can be changed.
- A hierarchy of active objects can be defined. The master active object is able to control the behavior of its subordinate active objects. Metaconstraints are used for this purpose.

3. Problem solving in TAO

TAO is implemented in the form of a constraint-based agent-oriented language. To specify an application problem in this language, the user has to perform the following three steps:

1) Decomposition of the problem into a collection of communicating agents.
2) Description of the structure and behavior of active objects. Any active object has slots whose values define its state, a model of behavior, and a set of references to external active objects. In the model, the slots of the active object and the slots of the external ones are connected by constraints.
3) Creation of the multi-agent system (TAO program). It is necessary to set the initial state of MAS and to specify the metaconstraints for control and reconfiguration. In fact, any complete TAO program can be regarded as a master agent with respect to the members of the implemented MAS.

Example. Consider the problem of supporting sustainable economic development of a region, e.g., Europe. The following sustainability criteria must be satisfied:

- the level of the economy in each country of the region should not fall below a certain limit;
- the development of the countries of the region should be smooth, without sharp falls and rises;
- the differences between the economic potentials of the countries of the region should be small, not exceeding a certain predefined limit.

When solving this problem with TAO, it is necessary to represent the countries of Europe by active objects. In addition, we introduce an active object representing an international bank providing loans to developing countries.

Each country is associated with a macro-economic model that links such parameters as national income, financial resources, social expenditures, investment in manufacturing, total debt, etc. TAO makes it possible to represent models of this type by algebraic equations and inequalities. The model includes also relationships determining the country's development in time.

Connections between active objects (countries) are determined by their import and export relationships. In addition, each country has relations with the international bank and can either increase its assets or take out loans on certain terms.

TAO allows us to link the model of a multi-agent system with its graphic representation. In Fig.1, we show the user interface of the system controlling the social and economic development of Europe [14].

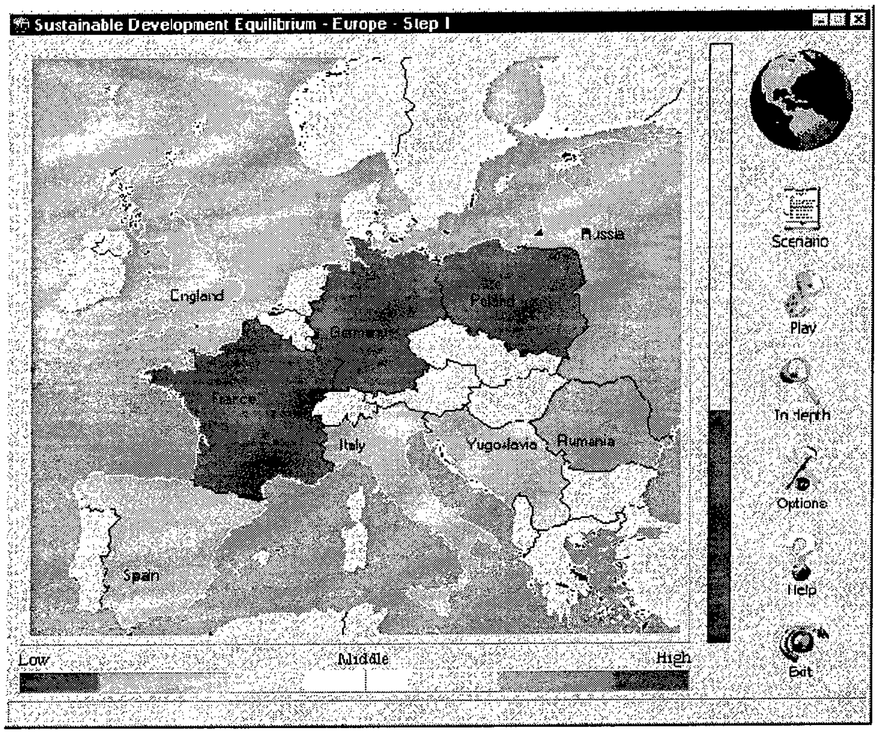


Fig.1. Supporting sustainable development of Europe

The countries are color-coded to reflect their stability level. The colors may change dynamically in the course of operation of the system. The closer a country's color is to green, the better the social and economic situation is in this country. Red colors correspond to crisis situations. One way to normalize the situation in a country is to get loans either from the international bank or from some other country. The user is playing the role of one of the agents of this MAS. His main task is to control the distribution of the resources of the international bank between the needing countries.

This multi-agent system is being developed in cooperation with FAW (Research Institute for Applied Knowledge Processing, Ulm, Germany) in the framework of the European project called ASIS (Alliance for Sustainable Information Society) [15].

4. Architecture of the TAO programming system

TAO is implemented in the form of an integrated programming system with a high-level input language used to specify multi-agent systems. The overall architecture of the system is shown in Fig. 2.

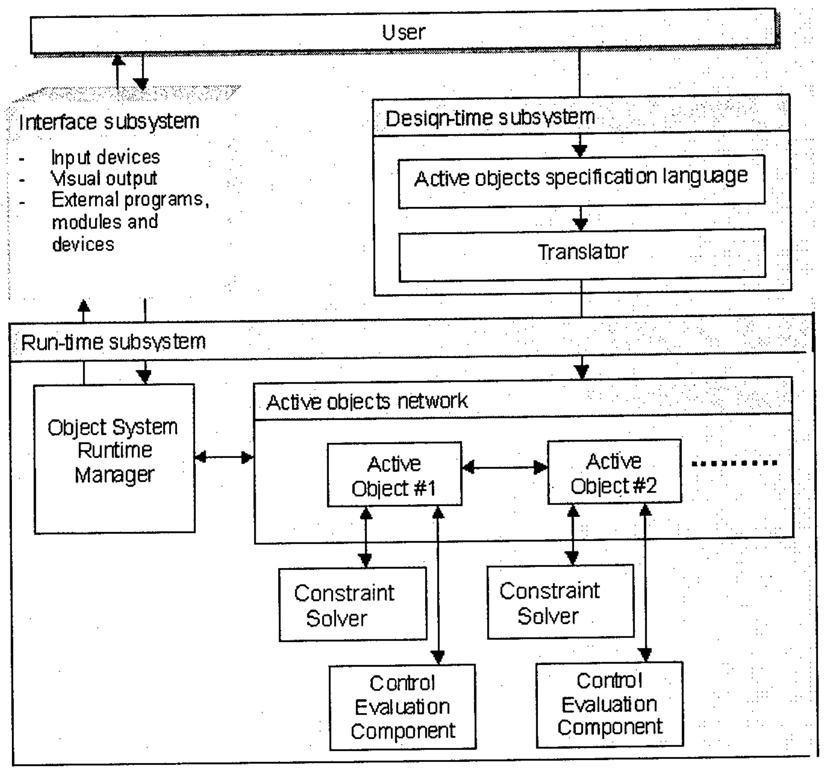


Fig. 2. Architecture of the TAO system

The main notion of the TAO language is that of an active object which has some properties of a traditional object; in particular, it can inherit information from other objects and ensures encapsulation of data and actions. We distinguish the following features of active objects:

- the behavior of an active object is described by a single model rather than by separate, unrelated methods;
- an active object is autonomous and exists over time.
- interaction between active objects is based on events rather than on exchange of methods as in the classical object-oriented approach.

The description of an active object includes, in particular, the following sections:

- a set of slots whose values determine the state of the active object;
- a set of references to external active objects whose state information this object can analyze;
- a set of local active objects that are subordinated to the current object;
- a model of behavior, represented by a system of constraints;
- control — a selected part of the model that is responsible for the reconfiguration of the local MAS.

On the whole, the MAS described in the TAO language is a hierarchic network of active objects. The compiler of the TAO system builds the executable representation of a network of active objects. This network may change dynamically in the process of MAS operation.

Since the states of active objects are calculated asynchronously, each of them can be associated with an individual software module that solves the corresponding system of constraints (the constraint solver). This module can be executed in a separate processor or node of the computational network.

Interaction between active objects is coordinated by a special manager. It ensures, in particular, it ensures exchange of data between active objects, receipt of signals and other information from the outside environment (including the user), switching the MAS to a new state, the interface with the

system performing graphic visualization of the behavior of active objects, etc.
In TAO, the user is regarded as an active object that can influence the operation of the MAS by generating signals, data, commands, etc.

5. Construction of dynamic multi-agent systems in TAO

The process of computation of the state of MAS is called in TAO a *computation step*. We assume also that there exists an abstract clock that counts the computation steps. In each step the environment goes into a new state. This means, in particular, that the state of MAS and, therefore, the states of all active objects have a fixed connection with the number of the computation step. The organization of calculations is such that the state of an active object can change at most once in each step. Each active object is recomputed at each step after the values of all of its outer objects have been calculated, and cannot change their values. To modify effectively the state of MAS, one needs additional information. Its sources in TAO are the following:

- the connection between active objects and the outside world;
- the dependence of objects on the previous state of MAS;
- the reconfiguration of the set of active objects.

Here the outside world is represented either by the user, who changes the state of MAS through input devices (keyboard, mouse, etc.), or some sensors or an arbitrary program (e.g., a DBMS), which informs MAS that new information has arrived. In all cases the connection with the outside world is provided by special procedures written in the TAO implementation language (C++). The interface to such a procedure can be described within an active object. Active objects that have a connection with the outside world are given a generic name *sensors*, since they generate events for the active objects. Sensors do not usually depend on other objects, and they are recomputed at the very beginning of each computation step.
We distinguish three classes of MAS with varying degree of dynamism: *static, with dynamic links and with a dynamic set of agents.* Each type is more general than the preceding one. We now examine each type in more detail.
Static systems. These are usually used in approaches related to describing individual agents [16] or stable systems [17]. The set of agents and the links between them are fixed prior to starting the system.
Systems with dynamic links. The most common and best-studied class of systems. The set of agents is fixed before the system is started, while the links between them are established and modified during its operation. A typical example is multi-agent systems in which the interaction between agents is based on a model of negotiation in its various forms: *private* (two participants), *tender* (one consumer, many suppliers), and *auction* (one supplier, many consumers).
Systems with a dynamic set of agents. The process of dynamic creation and destruction of objects is both simple and complex. Its simplicity is in the ease of its description within any approach (logic, functional, or object-oriented). Its complexity is related to correctly integrating a new agent in the system, or removing an agent without compromising integrity of the system. These problems can usually be solved in one-to-many negotiations (tenders and auctions). One can give two examples of such systems. The first example is HOMAGE [18], where an object is created by creating a new process with a special library function, and its integration in the system is described by the user through sending messages. Another example is Creché Simulation [19]. In this system, each agent is created with a certain information on its place in the community of agents; it has certain intelligence and a set of relevant knowledge. Since each agent is an element of collective intelligence, integration and removal of agents presents no problems.
In TAO for each active object, its *link* is represented by a set of other active objects that it can "see" at the time. If this set does not change over the entire lifetime of the system, we say that the link is static, otherwise the link is dynamic. On the one hand, each agent is independent; on the other hand, it is limited by its task and its model, which defines the types of other agents that the object can "see." The real agents existing near this object are shown to it by a certain management structure containing the object as its part. This combination of agents and the facilities for controlling them is called the *environment*. It defines the separate tasks for all objects and combines them all to solve one common problem.
We give below a fragment of the formal description of an environment.

Main (**Var** < *Environment variables* >
Agents < *The set of agents* >
Model < *Model of the environment* >
Init < *Initial values of variables* >
Control < *Control facilities* >
)

Like an active object, the environment contains a number of variables (or slots) whose values are recomputed in each step (section **Var**). The relationships between them are described in the form of a system of constraints in the section **Model**. The initial values are given in the section **Init**. The section **Agents** lists all agents of the system. We will not describe types of agents here; we will only say that their declarations should precede the description of the environment.

Finally, the **Control** section contains the program of the system's operation. It consists of a number of instructions that should be executed asynchronously. Stripped of the syntactic detail, an instruction looks like this:

<condition> → *<action>*.

Here *<condition>* means a certain predicate on the values of environment variables and the slots of agents that have been computed in the current step, and *<action>* refers to a certain operation (for instance, removing an agent or setting up a link between agents). As a result of applying an instruction, a certain configuration of agents and links is prepared for the next computation step.

We describe now the algorithm of operation of the entire system. In the preparatory step, all variables and slots are given initial values, and the initial links are created. Each computation step begins with recomputing all agents in the order determined by their dependencies on each other. Next, the model of the environment is recomputed. After that, the system is rebuilt by executing all instructions. It should be noted that the links between active objects can be changed only through explicit instructions. If new links are not specified for an agent, then it preserves its old links. After this section is executed, the system goes to the next computation step.

Let us now consider in detail how TAO constructs dynamic systems of classes mentioned above.

In static MAS, we define a set of active objects of certain types and the links between them which cannot change during operation of the system. An agent can change its state depending on the states of other agents. It should be noted that TAO already has certain dynamic properties at this level, provided by external stimulation through sensors.

In MAS with dynamic links we can modify the relations between agents during computations. Here all agents are known and described in advance, although they can leave temporarily the "field of vision" of the system. An example of such a system can be presented by a model of a family living in an apartment: we know all members of the family, but they are not always at home. They can enter or leave the apartment independently (they can go to work, to school, shopping, etc.). Each agent knows all others but does not always "see" them.

To support implementation of such systems, the definition language for multi-agent systems contains some new facilities in addition to the mechanism of link modification: the symbol NULL for an undefined object and the actions that remove or restore an agent. The *Restore* action indicates the appearance of an agent which had not been used in computations for some time. The *Remove* action removes the agent from the environment but not from memory; the values of its slots can be saved, to be restored at a later time when the agent returns to the environment.

MAS with a dynamic set agents can have an arbitrary number of active objects of a certain type which appear and disappear in a random manner. An example of such a system is a simulation of road traffic, where the number of automobiles and pedestrians is not known in advance. To implement such systems, one needs facilities for description of dynamic data structures and agent creation/removal. If there are many agents of one type and none of them has any advantages over the others, there is no sense in giving the agent a unique name. It is better to place it in a dynamic structure with standard access means. In TAO, such a structure is *list.* To *destroy* an agent, we remove it from the list and free the memory occupied by the agent. To *create* an agent, we allocate the memory and enter a reference to it in the corresponding list. The initial values of slots of the new agent are specified if necessary.

The above facilities are sufficient for implementation of complex dynamic multi-agent systems. In this section we consider the case when the system consists of agents of one level. On the other hand, the entire environment can be regarded as a higher-level agent whose behavior is described by the local multi-agent system. Continuing in this manner, we can construct hierarchic multi-agent systems.

6. Related work

Most multi-agent applications are presently developed with the help of technologies based on imperative programming languages, e.g., C, C++, or Java. These languages are extended by special facilities to support implementation of multi-agent systems. The main drawbacks of this approach are low-level, imperative style of system description. This increases development time and makes maintenance and further development more difficult.

Multi-agent technologies developed within the framework of AI allow one to create high-level, declarative descriptions of rather complex MAS. Intelligent agents have broad knowledge of the outside world, can adapt to changing environment, can set objectives, develop and implement plans to realize these objectives. To support these and other capabilities of intelligent objects, various AI techniques are used, e.g., production rules, temporal logics, fuzzy sets, constraint programming, etc. Two drawbacks of these complex intelligent technologies can be noted. First, they are complex to learn and apply. It is difficult for the user to master the whole variety of the proposed means and use them optimally. Second, most of these technologies are experimental and suitable only to develop demo systems, characterized by low reliability and efficiency.

The work in which constraint programming is combined with multi-agent technologies can be divided into two groups. In the first, agents are used to solve constraint systems. One of the best known representatives of this approach is Yokoo [20], who proposed the method of distributed constraint satisfaction. In this method, each variable is associated with an agent. These agents use a negotiating procedure to determine their states (values) satisfying the constraints. The emphasis in this type of research is on developing efficient algorithms for solving constraint systems, rather than on developing applied multi-agent systems.

TAO belongs to the second group that focuses on applying constraint programming techniques to implement multi-agent systems. One other method that is close to TAO is the distributed version of Oz [21], developed by G. Smolka and his colleagues in Germany. This very complex system integrates several paradigms: constraint programming, logical programming, and functional programming. Oz supports working with Internet and can be used to develop physically distributed multi-agent systems.

Unlike Oz, TAO has the following properties:

- All aspects of agent operation are described in a single manner, with constraints. This ensures ease of learning and using TAO compared with other technologies;
- TAO has facilities for working with time, which significantly expands its area of application;
- The SD-method used in TAO to solve constraint systems makes it possible to work efficiently with agents possessing very complex behavior models that cannot be realized in other systems.

Thus, TAO is a high-level programming technology that allows one to create complex (including intelligent) and efficient MAS for a broad range of applications.

7. Conclusion

The technology of active objects belongs to a new generation of tools providing comprehensive support to the development of multi-agent systems. It extends the capabilities of the conventional object-oriented approach and integrates it with such promising AI methods as dynamic constraint programming and distributed knowledge processing systems.

A principal novelty of TAO is that it uses constraint programming methods to specify all aspects of behavior of multi-agent systems, which allows one to simplify their design and development. The computational capabilities of TAO are based on the method of subdefinite models, which has been developed in our Institute and is presently recognized as one of the most powerful and efficient approaches in constraint programming.

The multi-agent technology that we propose can find application in various fields, for example:

- distributed control of technological processes;
- supporting environmental, economic, social and other types of monitoring, jointly with other components of geoinformation systems;
- development of optimal scenarios for military and rescue operations;
- modeling complex processes and technical devices for design, diagnostics, and training;
- specification of the behavior of autonomous mobile robots working in a dynamically changing environment.

References

1. Wooldridge, M., and Jennings, N.R. Agent Theories, Architectures and Languages: A Survey. ECAI-94 Workshop on Agent Theories, Architectures and Languages, Amsterdam, Netherlands, 1994: 1-39.
2. Adamatzky A., Holland O. Swarm Intelligence: Representations and Algorithms. Proceedings of International Workshop on Distributed Artificial Intelligence and Multi-agent Systems - DAIMAS'97, St. Petersburg, June 1997, pp.13-22.
3. Rao A., Kinny D., Georgeff M. A Methodology and Modeling Technique for Systems of BDI Agents. In Proceedings of 7th European Workshop on Modeling Autonomous Agents in Multi-agent World. Number 1038 in Lecture Notes in AI. Berlin 1996, pp. 56-71.
4. Fonseca J., Oliveira E., Steiger-Garcao A. MACIV — A DAI Based Resource Management System. Proceedings of PAAM'96, London, UK, April 1996, pp. 263-277.
5. Jörg P.Müller. The Design of Intelligent Agents: a Layered Approach. Lecture Notes in Artificial Intelligence, vol. 1177, Springer-Verlag Berlin Heidelberg, 1996.

6. Narin'yani. Subdefiniteness in knowledge representation and processing, Transactions of USSR Acad. Of Sciences, Technical cybernetics, Moscow, No. 5, 1986, 3-28 (in Russian).
7. Narin'yani, A.S.; Borde, S.B.; and Ivanov, D.A. Subdefinite Mathematics and Novel Scheduling Technology. *Artificial Intelligence in Engineering* 11, Elsevier Science Limited, 1997.
8. Zagorulko Y., Popov I. Object-Oriented Language for Knowledge Representation Using Dynamic Set of Constraints. Proc. of the Joint Conf. On Knowledge-Based Software Engineering in Smolenice, Slovakia, 1998, pp. 124-131.
9. Shvetsov I., Semenov A. & Telerman V. Application of Subdefinite Models in Engineering. Artificial Intelligence in Engineering, N 11,vol 1. Elsevier Science Limited, Great Britain, 1997, pp. 15-24.
10. Babichev, A.B.; Semenov, A.L.; et al. UniCalc, A Novel Approach to Solving Systems of Algebraic Equations. In Proceedings of the International Conference on Numerical Analysis with Automatic Result Verifications. Lafayette, Louisiana, 1993.
11. Shvetsov I., Nesterenko T., Starovit S. Integrating Constraint Programming and Agent-Based Technology. Proc. of International Workshop on Distributed Artificial Intelligence and Multi-agent Systems - DAIMAS'97, St.Petersburg, June 1997, pp. 213-222.
12. Shvetsov I., Nesterenko T., Starovit S. Technology of Active Objects. Proc. of AAAI Workshop 'Constraints and Agents', Providence, USA, July 1997, pp.76-84.
13. Shvetsov I., Nesterenko T., Starovit S. TAO: A Multi-agent Technology Based on Constraint Programming. Sixth Scandinavian Conference on Artificial Intelligence - SCAI'97, IOS Press, Helsinki, Finland, August 1997, pp.151-161.
14. Hayes-Roth. An Architecture for Adaptive Intelligent Systems, Artificial Intelligence: Special Issue on Agents Interactivity, (72): 329-365, 1995.
15. Levin D. Computer Games and Ethical Foundation of Sustainable Development. Environmental Informatics: Information and Communication Technology for Sustainable Development.Vol.4, Moscow-Novosibirsk-Ulm, 1997, pp.44-54.
16. Environmental Informatics: Towards a Sustainable Information Society. Vol. 5, Moscow-Novosibirsk-Ulm, 1998.
17. Michael Schroeder, Iara de Almeida Móra, Luíz Moniz Pereira. A Deliberative and Reactive Diagnosis Agent Based on Logiç Programming, Intelligent agents III – Proceedings of the Third International Workshop on Agent theories, Architectures and Languages (ATAL-96), Springer-Verlag, Heidelberg, 1996.
18. Agostino Poggi, Giovanni Adorni. A Multi Language Environment to Develop Multi Agent Applications, Intelligent agents III – Proceedings of the Third International Workshop on Agent theories, Architectures and Languages (ATAL-96), Springer-Verlag, Heidelberg, 1996.
19. Darryl Davis. Reactive and Motivational Agents: Towards a Collective Minder Intelligent agents III – Proceedings of the Third International Workshop on Agent theories, Architectures and Languages (ATAL-96), Springer-Verlag, Heidelberg, 1996.
20. Yokoo, M. Distributed Breakout Algorithm for Solving Distributed Constraint Satisfaction Problems. Proceedings of the Second International Conference on Multiagent Systems(ICMAS-96), pp.401-408, 1996.
21. Haridi, S., Van Roy, P., and Smolka, G. An Overview of the design of Distributed Oz. In the 2nd International Symposium on Parallel Symbolic Computation (PASCO 97). ACM, New York, 1997.

CONTENT BASED PARALLEL IMAGE RETRIVAL WITH MULTIAGENT COOPERATION

Zhongzhi Shi, Hu Cao, Wei Wang

{shizz, ch, wei}@ics.ict.ac.cn
Institute of Computing Technology,
Chinese Academy of Sciences
Beijing 100080, China

Abstract

Image can be retrieved with the combination of features, such as color, texture, and shape. The prevailing content based retrieval method is summing the weight retrieval result of individual feature, which can just roughly reflects user's query description. We present a parallel image query representation, which adopts tree to describe user's query. By searching individual R+ trees of color blocks, texture, and shapes, a cooperative algorithm among individual query agents with different features is designed coordinate them and to speed up the retrieval process. With a multiagent-developing tool AOSDE built by us, a prototype system is developed and some benefits on representation of user interest and speed are showed in the results.

Keywords: *Image Retrieval, Multiagent System, Multiagent Cooperation*

1. Introduction

Image Information is widely used in the some important applications, such as digital library, art works collection and museum management, geographic information system, remote sense and earth resource management, weather forecast, and fashion design. The researches on building and search image database follows two major approaches: one method identifies image content with text then search image with the regular function of relational database; The other method searches image with basic statistical image features such as color, texture, and shape. These two approaches are complemented and can be used together.

Plenty of works have been done using searching with individual feature. But these features themselves can not express our cognition of image. People percept an image as a group of regions, inside which there is some similarity of color and texture. The shape of an image is described by the boundary of these regions and their spatial relation. Describing an image query and search with the combination of color, texture, and shape can define user's search request more concretely therefore retrieval results is a smaller and accurate set of images to represent user's interest. QBIC, Virage, and VisualSEEK are three systems that synthesize two or three features into their query algorithms[1][7][8]. Developed by IBM Almaden research center, QBIC system (Query by Image Content) use color and texture pattern, spatial description, and text information to query image database. Similar to QBIC, Virage, a content based search engine by Virage, support visual query based on color, color layout, texture, and structure. User can combine these atomic queries arbitrarily by adjusting the weight of queries. VisualSEEK studies the spatial relation query of image regions and visual feature extraction from compressed format. The visual features adopted by VisualSEEK are color set and wavelet-based texture.

The methods used in these systems just provide a way for user to select retrieve schemes. The similarity between query request and images in database is assigned with the weight sum of every individual retrieval result, where the weights are set by user according to his(her) fondness of features. For example, by selecting large weight on color and small ones on texture and shape, the result of color based query will play a major role in deciding the final result image set. But simply summing the result can not sort out of the most similar image in some situations for the spatial relation among feature is omitted. Keeping the other conditions constant, the image with greater similarity on one feature will rank higher in final result image set. Sometimes the features scatter through that image, which means that all the features exist in a single image. While as we often want to identify an object in the image, where these features should located in one region of the image. So that the result of retrieval does not reflect the user query interest loyally.

When users want to retrieve an image by describing an object that should be in the result images with the basic statistic features, the query is a kind of

spatial "And" of atomic queries on every features, i.e. all the feature regions must locate in roughly the same part of image. To utilize the existing retrieve technique on color, textures, and shape, a coordination and negotiation process is need to synthesize each query's partial results and synchronize the search progress. When one query agent find a feature region in the image, it should call other agents to search this area to test the occurrence of other kinds of features in this smaller region. If this area contains all features defined by user judged by some criteria, the image is one of those that meet the query standard; otherwise, we must further search the image to decide whether it contains the queried object.
A R+ tree scheme is introduced in this paper to represent the spatial dimension of a feature. Based this representation, a search algorithm for individual features in presented. With this algorithm, the image can be searched in parallel using color, texture, and shape. A cooperation mechanism is designed to coordinate to retrieval process. When one significant feature in one region is found by an agent, some notification messages will be sent to other agents to inform them the high possibility of the occurrence of the object to be searched in the region. The similarity of an image is decided by the spatial distance weight feature distances of color, texture, and shape. A content based image retrieval system is designed, which consists of three search agents, a task allocation and result synthesizing agent, and a browser to accept user query condition and display the final result. After receiving query request from browser, the task allocation and result-synthesizing agent allocate three atomic queries to search agents. Using color, texture, and shape, these search agents compare the images in database with atomic features then summit partial results to task allocation and result synthesizing agent. This agent selects out of a set of most similar images by the synthesized result. The final retrieval work is carried out by human being to further analyze this image set and select the best images. The system works in parallel and asynchronously to speed retrieval.
The rest of the paper is organized as follows. In section 2, we introduce R+ tree for image representation. In section 3.1, the calculation of the similarity of an image with query condition and a search algorithm with one feature are presented; based on this algorithm the parallel retrieval with multiple features is given in section 3.2. In section 4, we introduce a prototype agent based parallel image retrieval system. Finally, we give the conclusions of this paper, which discuss the benefit and limit of our works.

2. The R+ tree of image feature regions

We can consider an image as a set of regions. A region is a connected set of pixels that are homogeneous with respect to a criterion of homogeneity. An example of such a criterion is that the maximal difference of gray values or colors of the pixels is below a threshold. Various heuristics have been designed to guide the determination of regions. This issue will not be discussed here.
To deal with the great number of images in the images, the use of R+ tree provides an efficient access method. After segmentation of regions, these two dimensional objects are simplified as the minimal rectangles that enclose the objects. The leaf node of R+ tree can be defined as a set of (oid, RECT), where oid is the identifier of the object in the database. RRCT is quadruple (Xlow, Xhigh, Ylow, Yhigh), which describes the boundaries of the minimal rectangle; Xlow, Xhigh is lowest and most upper X-coordinate of feature region, and Ylow, Yhigh is lowest and most upper Y-coordinate of feature region. The non-leaf node is defined as (P, RECT) set, where P is the pointer to lower level nodes in the tree. The node A has a pointer P to node B means that region B is enclosed in region A or have same overlapped area with region A. As illustrated in Figure 1. The objects in the figure construct one R+ tree shown in the right side of the Figure1.

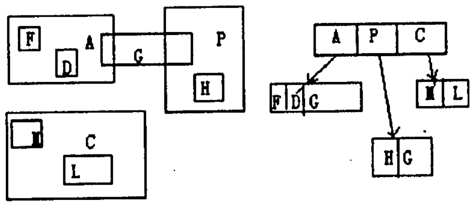


Figure 1. Some two-dimensional spatial objects represented as a R+ tree.

R+ tree has the following properties:

- For every item of middle nodes, (P, RECT), the root of subtree pointed to by P has rectangle R if and only RECT encloses R. An Exception is that R is a rectangle of leaf node. At this time, R can be enclosed by RECT or intersect with RECT.
- The Intersection of arbitrary two items of one middle node, (P1, RECT1) and (P2, RECT2), is an emptiness set.
- Root has at least two sons, unless it is a leaf itself.
- All leafs are in the same level of a R+ tree.

In our retrieval system, a color image can be understood as follows: Suppose there is a color image shown in figure 2

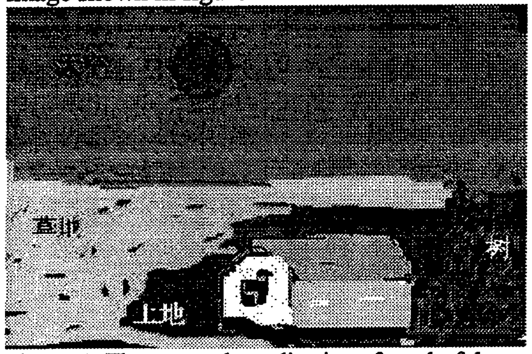
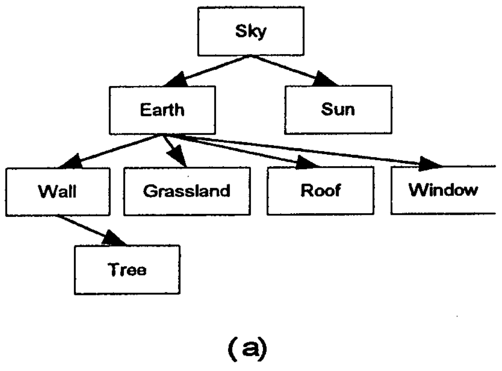


Figure 2. The grayscale replication of a colorful image with sky, sun, grassland, tree, and a house

When we use feature color to build the R+ tree, Figure 2 consists of: 1. Sky(blue); 2. Sun(red); 3. Grassland(reseda); 4. Tree(bottle green); 5. Roof (yellow); 6 Wall (gray); 7. Window(black); 8. Earth(brown). With different understanding to the image, two R+ trees can be constructed: (1). The R+ tree of the image is constructed by pasting small color blocks upon large ones, where the large color block has a pointer to small ones. First, draw rectangle blue sky; second, paste the red sun in the sky and the brown earth in the lower part of sky; then paste the grassland, tree roof and wall in the earth block, finally, the windows is pasted on the wall. (2). We paste small color block upon large color block only when it is enclosed by the large one. So that only sun and window is pasted and the other are parallel. In this situation, setting a virtual root is necessary to construct the R+ tree. These R+



trees are shown in Figure 3.

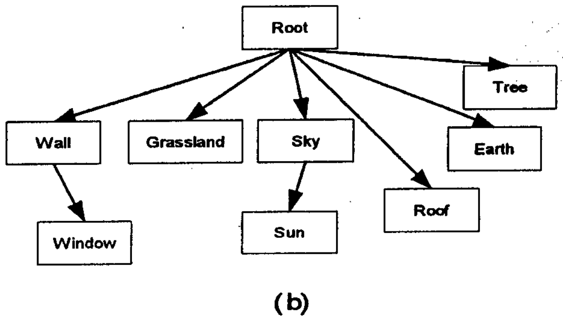


Figure 3. Two R+ trees built with different understanding to the Figure 2. (a) is construct with the first kind of understanding; (b) with the second one.

3. Parallel Retrieval with Color, Texture, and Shape

Image Retrieval is to find out the images that are similar to a provided image in an image database. With the R+ tree representation for an image, the retrieval can be carried out as search of a R+ tree.

3.1 Image Retrieval by R+ tree search

To identify a person or an object in an image is an extremely hard work, but to find out a similar image is not so formidable. The similarity of two images can be described using the feature distance between them. As an image is determined by feature $X = (f_1^X, f_n^X, \ldots, f_n^X)^T$, the similarity between image $A = (f_1^A, f_n^A, \ldots, f_n^A)^T$ and image $B = (f_1^B, f_n^B, \ldots, f_n^B)^T$ can be calculated by the feature distance as follows:

$$dis_f(A,B) = \sum_{i=1}^{n} \omega_i (f_i^A - f_i^B)^2 \tag{1}$$

Where $\omega_i (i = 1, \ldots, n)$ is the weight of microfeature i, which determines the importance of this microfeaure. The larger is $dis_f(A,B)$ the more similar are A and B. Euclidean distance is not a good measure for perceptive space distance. However, It is hard to find out another measure before we study the human vision and perception mechanism deeply. Consider that the emphasis of the paper focus on mutliagent cooperation, the prevailing Euclidean distance is adopted in this paper.

Color, texture, and shape are three often used features to calculate the feature distance of an image. Color can be represented by RGB standard where R stands for red; G stands for green; B stands for blue. The distance between two color blocks can be defined as

$$dis_{COLOR}(A,B) = \int_R \sum_{i=R,G,B} w_i \left(f_i^A - f_i^B\right)^2 dxdy \tag{2}$$

Where R is the overlapped region of two color blocks, and f_i^A and f_i^B are one RGB value of pixel (x, y). Texture feature distance between image A and image B can be defined as

$$dis_{TEXTURE}(A,B) = \sum_{i=1}^{4} \omega_i (q_i^A - q_i^B)^2 \qquad (3)$$

Where q_1, q_2, q_3, q_4 are second order moment, contrast, entropy, and correlation texture feature of an image, $\omega_i (i = 1,...,4)$ is the weight of these microfeatures. Because complex shape is difficult to be described and matched, simple shape templates are used. The shape feature distance can be determined by some simple features, such as size, perimeter, and shape, etc. But this measure sometimes has great error, some methods based on template distortion should further used to correct the error[3], which will not be detailed in this paper. In an image retrieval system, user usually queries through providing a small piece of image or describing the features of the image, which can be converted into a R+ tree. Using this R+ tree, we can use deep first search or breadth first search to match two R+ tree, then calculate the feature distance. The whole process of image retrieval with single feature is described as follows:

ALGORITHM 1:

```
Step 1. Using the interactive graphic user interface provided for user, user generates a search tree R_1 that is a description of the desired image.
Step 2. Unsearched_image_set = Image database;
Step 3. While (Unsearched_image_set is not empty) do{
    R_2 = an image in the unsearched_image_set;
    Unsearched_image_set=Unsearched_image_set-R_2;
    Search R_2 with R_1, and calculate $dis_f(R_1, R_2)$;
}
Step 4. Return some images that have minimal feature distance.
Step 5. If user does not satisfy with the search research result, then:
    (1). Generate a new search tree, go to step 2. or
    (2). Use one of the result images as the template to generate a new R+ tree, go to Step2.
Step 6. If User satisfies with the search research result, then return.
```

3.2 Parallel Retrieval with Color, Texture, and Shape

Multiple features can better describe an image than single one, especially when they are defining the same object in the image. Since these features must have spatial correspondence in the image, the total feature distance is determined by the individual feature distances and the spatial distance among every feature regions. If color, texture, and shape are used to describe an image, and R_C, R_T, R_s are three rectangles of these features in the R+ tree, we can use the distance between the barycenters of those regions and the barycenter of them to define the possibility that the region are the features of the same object. Hence the total feature distance of color, texture, and shape is:

$$dis(A,B) = \sum_{i=C,T,S} w_i dis_i \left| \vec{C_i} - \vec{C} \right|^2$$

(4)

where $\vec{C_i}$ is the barycenter of one feature region i, $\vec{C}$ is the barycenter of all the regions, and dis_i is the feature distance of feature, $C \in COLOR, T \in TEXTURE, S \in SHAPE$ are color, texture, and shape region respectively, and COLOR, TEXTURE, SHAPE are set of all regions of every feature R+ trees. The total feature distance of the image is

$$dis(A,B) = \min_{i \in COLOR \times TEXTUREXSHAPE} \{dis_i(A,B)\} \quad (5)$$

Because it is necessary to traverse three R+ trees and find the feature regions set of these features, and sort out the minimal distance from the cartesian set of those features, the calculation of minimal feature distance is a time-consuming task. However, we need not to know the minimal feature distance since we just want to identify the existence of some combined features in the image. Usually, user make the final selection from an image set, whose elements are selected out for their feature distance is less than a threshold T:

$$\mathrm{Re}\,sultSet = \{i \mid i \in imagebase \wedge (dis_i < T)\} \quad (6)$$

So it is often unnecessary to calculate the minimal feature distance of an image, instead using the heuristic that there must have three "salient" features is the same region, we can redefine (6) as $i \in \mathrm{Re}\,sultSet$ iff

$$\bigvee_{\substack{l \in COLOCR \\ m \in TEXTURE \\ n \in SHAPE}} (E_l(C) \,\overline{\wedge}\, E_m(T) \,\overline{\wedge}\, E_n(S))$$

(7)

where E_i (X) is given as the existence a feature x in a region i by

$$E_i(X) = \begin{cases} 1 & if\ dis_X < Threshold_X \\ 0 & otherwise \end{cases} \qquad (8)$$

$\overline{\wedge}$ is a spatial AND operator that denotes two regions are the same part of the image, where the ratio of the size overlapped part of region A and region B and the size of region A and region B is adopted

$$A \barwedge B = \left(\frac{overlappedsize(A,B)}{size(A)} > C\right) \wedge \left(\frac{overlappedsize(A,B)}{size(B)} > C\right) \quad (9)$$

Where C is a constant. The position information can be obtained from the RECT item of the nodes in R+ tree.

To speed up the retrieval process, we can search the image by color, texture and shape in parallel. In the parallel search process, if a "salient" feature is found, i.e. its feature distance is greater than a threshold, some notification should be sent to other search process to pay attention to the same region in their R+ tree. Using information of feature rectangle, we can break the other search processes, and find out the spatial corresponding feature regions in the other R+ trees by looking up the RECT entries of each element of nodes. Then the feature distance of other two features can be found. If these two feature distances are also "salient" enough, we can use (5) to calculate the feature distance of the image; otherwise, all search agents continue their searches. This process will be detailed in the next section. When algorithm 1 is used parallel retrieval, the step 3 in algorithm 1 should be modified as follows:

```
Step 3'. While (Unsearched_image_set is not empty) do{
R_2 = an image in the unsearched_image_set;
Unsearched_image_set=Unsearched_image_set–R_2;
While(Search R_2) do{
    If (a feature region is found){
        notify it to other agents;
        continue;
    }
    if (get a message to look up a region){
        find the region;
            return the feature distance of this
        region to result synthesizing agent;
    }
    if (get a break message)
            break the search;
    }
}
```

4. Agent Based Parallel Image Retrieval System

From the analysis of the preceding section, we have developed a prototype agent based parallel image retrieval system using the idea mentioned in section 3.2, which is a subsystem of Multimedia Information Retrieval System(MIRES)[4]. This system consists of: agent facilitator, color retrieval agent, texture retrieval agent, shape retrieval agent, task allocation and result synthesizing agent, browser, and image database. The image to be retrieved is stored and indexed in the image database. Agent Facilitator provides some system-level services, such as naming service, query service, subscribe service, and agent lifecycle service to facilitate the communication and cooperation among the functional agents. Feature retrieval agents have the capability of search the image base with one kind of feature. Color retrieval agent queries the image atabase by color; Texture retrieval agent queries the image database by texture; And shape retrieval agent queries the image database by shape. Since no agent has sufficient competence, resource, and information to solve the entire problem. A central controller is needed. The task allocation and result synthesizing agent receives query conditions from browser and allocate query task to the three search agents; this agent also store the temporary search results from the search agents and return the final result to the browser, which is the user interface of the system. The architecture of the whole system is shown in Figure 4.

In this system, agents communicate each other with a subset of KQML performatives. The agents are developed with one multiagent system developing tool AOSDE[5][6]. As illustrated in Figure 5, Each retrieval agent consists of five components: generic agent template, communicate module, interface, search engine, and user interface. Generic agent template provides the some generic abilities and knowledge for agent, such as communication ability with an agent communication language, the knowledge on interaction with agent facilitator, etc. Communicate module deals with the physical issues of agent communication and provides message buffers to meet the KQML specification's requirement on agent communication[2]. Search engine is the some pre-built 'legacy system', which is connected with an interface that exchange function calling messages and data between generic agent template and search engine. Although it is not obligatory, user interface gives some tracing and debug information to facilitate system debugging.

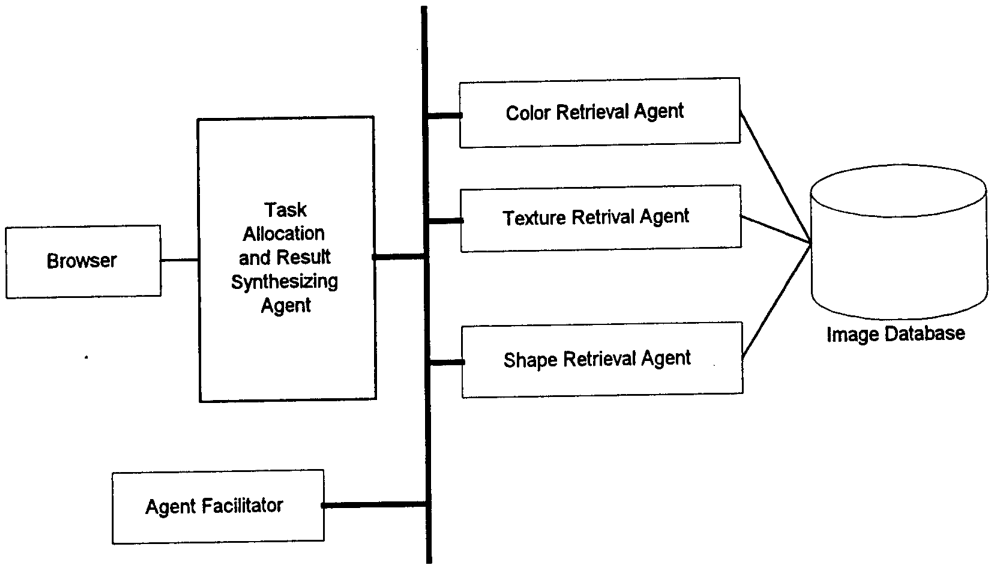


Figure 4. Architecture of Agent Based Parallel Image Retrieval System.

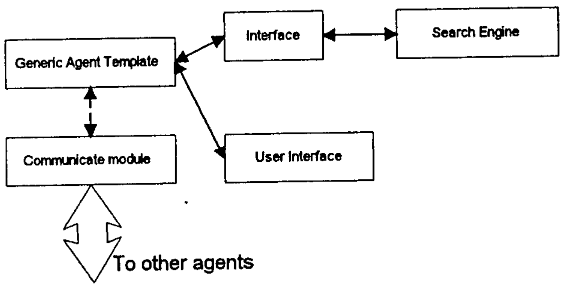


Figure 5. The Structure of Search Agent.

The image retrieval is carried out by the cooperation among task allocation and result synthesizing agent and retrieval agents. The mechanism is depicted in Figure 6. There are a task allocation module and temporary partial result table in task allocation and result synthesizing agent. The task allocation module will assign tasks to the retrieval agent when a user request a query or some partial result is gotten. The temporary partial result table stores the partial query results returned from retrieval agents according to image and task. It also reminds task allocation module to assign retrieval agents to search a specific region when a partial result arrives. When all the partial results have been returned, it return the total feature distance of the image to browser and tell task allocation module to notify the retrieval agents that this image is done and assign new task to them. The tasks assigned by task allocation agent are stored in task queue of the retrieval agent. There are three kinds of tasks: search an image with feature, look up one feature region, and clear all the tasks of one image. Task monitor is activated when a new task is arriving or the search engine is idle. If the task on the queue is to search an image with feature, it will tell the search engine searching the image with algorithm 1. If the task is to look up one region, search engine will search the image with spatial information in the R+ tree. If the task is to clear the tasks of the image, it usually means the similarity of this image has been calculated. So task monitor will clear those tasks and notify search engine to forget the current task and set the state of search engine as idle. Search engine searches the image space, and if it find a "salient" feature, returns it as the partial results to result synthesizing agent.

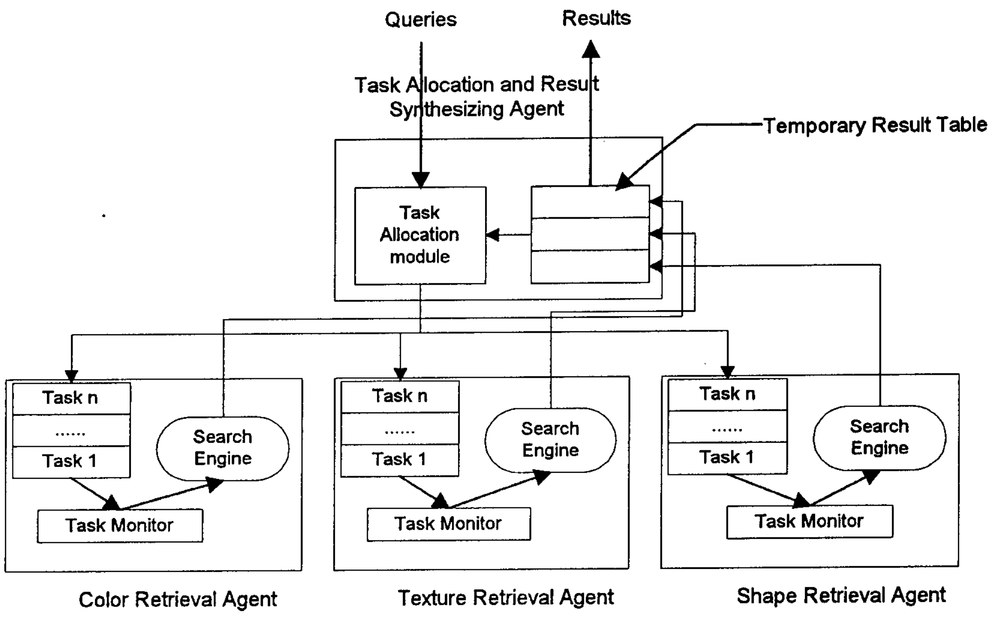


Figure 6. Cooperation mechanism among Agents. (Notice that although some lines connect the structures in agent directly, agents communication with other using a subset of KQML).

With such a mechanism, the image retrieval process can be speeded up in the following three manners:

- The retrieval processes in parallel. Running on the network environment, the system uses three computers as the retrieval agents. The increase of computing power shortens the retrieval time.
- The retrieval processes asynchronously. When an agent have finished search of one image, it can search another image, instead of waiting other agents to finish their works.
- Some heuristic on spatial correspondence of features is used to limit the search space. When an agent finds out the feature regions having salient feature, other agents can just search these regions instead of traversing the whole space.

5 Conclusions and Future Works

In this paper, We present an agent based parallel image retrieval system, in which the three search agents which search the image as a R+ tree in parallel are coordinated by a task allocation and result synthesizing agent. Since multiple spatial corresponding features are used in the parallel retrieval, user can more accurately express his or her query interest. The final search result of these features reflects both the significance of these features in the image and the spatial intersection of them. With the task schedule function of the task allocation agent, the parallel searches process asynchronously in limited search space, therefore it reduce the retrieval time.

There are still some problems that should be resolved in our future works. The relationship between feature and human cognition should be deeply studied to describe user's query more accurately. Often the cognitive expression on an object is difficult to be described by user with the features used in our system, i.e. color, texture, and shape. But user can tell his or her query condition using high-level concepts easily. So some research on human cognitive expression is proceeding. Another work is to study the method of the partition of texture and region, which effect the retrieval result greatly.

Acknowledgement

This research was supported by the '863 National Hi-Tech program of China. (863-317-9604-05)

Bibliography

1 Faloutsos M., Flickner W., Niblack D., Petkovic W., Equitz R., Efficient and effective querying by image content. Technical Report, IBM Research Report, 1993.

2. Finin T., Labrou Y., etc. KQML as an Agent Communication Language. In software Agents, edited by Bradshaw J. M.,AAAI Press/The MIT Press. pp.291-316, 1997.

3. Jimin Liu, Wei Wang, Zhongzhi Shi, A new deformed template match method, Chinese Journal of Computer(In Chinese), April, 1999.

4. Zhongzhi Shi, The Research and Developing of Feature Based Multimedia Information Retrieval System MIRES. Technical Report, 1998.

5. Zhongzhi Shi. Agent Oriented Software Developing Environment—AOSDE. Technical Report, 1998.

6. Zhongzhi Shi, Hu Cao, Yunfeng Li, Wenjie Wang, Tao Jiang, A Building Tool for Multiagent Systems: AOSDE. IT \& Knows, IFIP WCC '98, 1998

7. John R. Smith and Shih-Fu Chang, VisualSeek: A fully automated content-based image query system, In Proc. ACM Multimedia 96, 1996.

8.http://www.virage.com

ORDER CO-ORDINATION IN MANUFACTURING NETWORKS THROUGH A MULTI-AGENT-SYSTEM[1]

W. Sihn, H.-M. Dudenhausen

Fraunhofer Institute Manufacturing Engineering and Automation
Nobelstraße 12
D-70569 Stuttgart

Abstract

Enterprises co-operate with one another and form manufacturing networks in order to benefit from market potentials which they could otherwise not reach on their own. The operation of manufacturing networks, however, is not supported properly by existing production planning and control (PPC) systems. These systems allow neither communication with systems of other enterprises, nor the synchronisation of master production schedules in manufacturing networks. This paper proposes an order co-ordination method based on automatic negotiations in a Multi-Agent-System. This system allows the inclusion and scheduling of orders in real time within manufacturing networks in the semiconductor industry and, at the same time, co-ordinates the individual master production schedules (MPS) with one another.

Keywords: *manufacturing on demand, production planning and control, manufacturing network, genetic algorithms*

1. Introduction

The ability to react quickly to changes is a deciding factor for the success and survival of an enterprise in today's competitive market. Within the framework of Production Planning and Control (PPC), competitiveness depends on immediate fulfilment of customer desires, continuous supervision of goods and service, production processes, undelayed recognition of emerging conflicts, and quick responses in order to minimise any negative consequences. These abilities provide not only the basis for the successful operation of individual businesses, but also production networks. However, current PPC systems are not able to sufficiently support the operation of production networks. They do not allow for the communication between several enterprises' PPC systems, nor the synchronisation of the individual network partners' production plans.

Due to the strong logistical interdependence between plants, separate production planning for each plant generates only unsatisfying results. A lack of coordination and smoothing leads to overstock, high lead times, and deficient customer orientation (Arnold, *et. al.*, 1996).

In order to operate production networks, order management systems are needed, which streamline and monitor the production of goods and services of every unit within a production network along the entire process chain (Dudenhausen, *et. al.*, 1996). Within the scope of order coordination, the order flow through the production network has to be planned, controlled, and monitored. Cost aspects, such as high machine utilisation and low inventories, as well as customer satisfaction have to be taken into account. Besides low prices and high product quality, short delivery times and delivery reliability contribute significantly to customer satisfaction. Due to legal independence and the resulting variety of interests in production networks, the solution of goal conflicts (which can be rather difficult in individual businesses) is highly complicated in these networks.

[1] The development of this system is sponsored by the European Union within the framework of the ESPRIT project 20544 X-CITTIC in which, besides the IPA, the firms Alcatel Mietec, GEC Plessey Semiconductors, TEMIC, Nimble, and the research institutes Imperial College and INESC participate.

2. The Tasks of Order Coordination

Order coordination includes both the proactive abilities, such as

- order clarification
- order rough planning (rough scheduling and overall resources planning)
- submission of offer
- confirmation of order

as well as the reactive functionalities, such as

- order control
- reactive planning.

Order clarification includes the examination of the technical feasibility of the desired product (Kurbel, 1993). Within the scope of order rough planning, a cost-efficient solution for order fulfilment has to be determined (Westkämper, *et. al.*, 1997). With regards to the current state of the individual network units, which differ with respect to practicable operations, available capacities, lead times, cost structure, geographical location, and inventories of unfinished and finished products, an optimised route through the network has to be chosen for the potential order. The route through the network determines which units have to produce what quantities of which (intermediate) products within what time frame. Figure 1 shows such a route through a network. The starting points of the line graphics representing the route show in which units production is started and from which location intermediate products are taken.

Once the customer order is accepted, a comparison of order data and offer data is initiated (Hornung, *et. al.*, 1996). If they match, all network units participating in the completion of the order receive suborders (according to the offers they have made) for the services they have to render. In case they do not match, a new rough planning has to examine if the order is to be rejected or accepted despite the discrepancy. Figure 2 summarises the processes of the order co-ordination's proactive tasks.

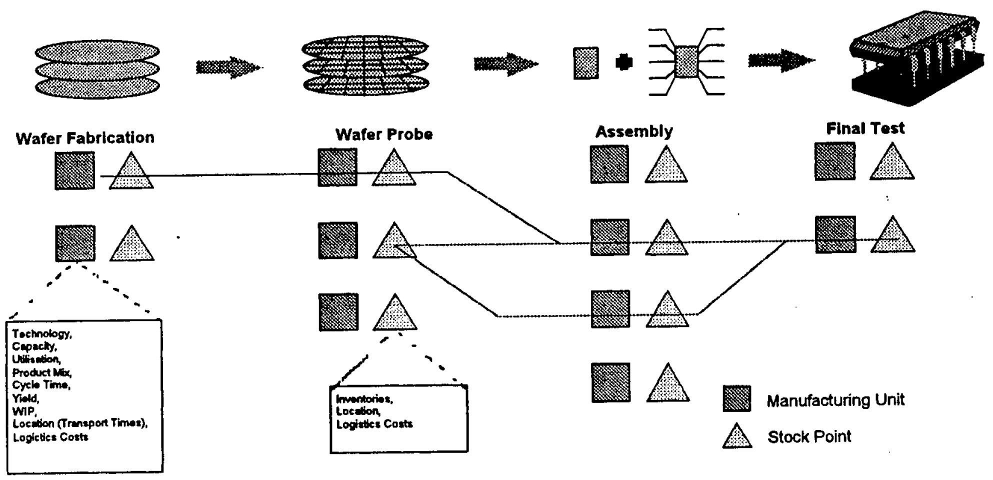


Figure 1: Optimised route of an order through a VE

The central problem of order coordination is the order rough planning. Within the scope of this rough planning, the partners have to agree on the submission of a common offer and the possible completion of the order. The submission of a binding offer requires all network units participating in the completion of the potential order to give their consent. Part of the agreement to submit an offer held by all partners is to assign subtasks to the respective network units, set the dates for these subtasks and the prices partners ask for services. These decisions cannot be made independent of each other. Therefore an order rough scheduling and an order-specific resource rough planning has to be carried out simultaneously with the submission of the offer and the determination of delivery dates.

An agreement can be reached either previously through the conclusion of basic contracts or through individual negotiations for each order. Individual negotiations make it more difficult to meet the customer demands for immediate acceptance of orders and quick responses to inquiries concerning delivery conditions or unused capacities.

Following the proactive order scheduling, it is the task of the reactive order coordination to ensure implementation of the plan by means of the two

main functionalities, i.e. order control and reactive planning.
It is the task of order control to supervise production processes and the initiation of reactive planning and control measures in case of missed deadlines. The order control's subtasks include the monitoring of the order progress, the forecast of finish dates and quantities, as well as the comparison of desired and actual values in order to recognise plan deviations.

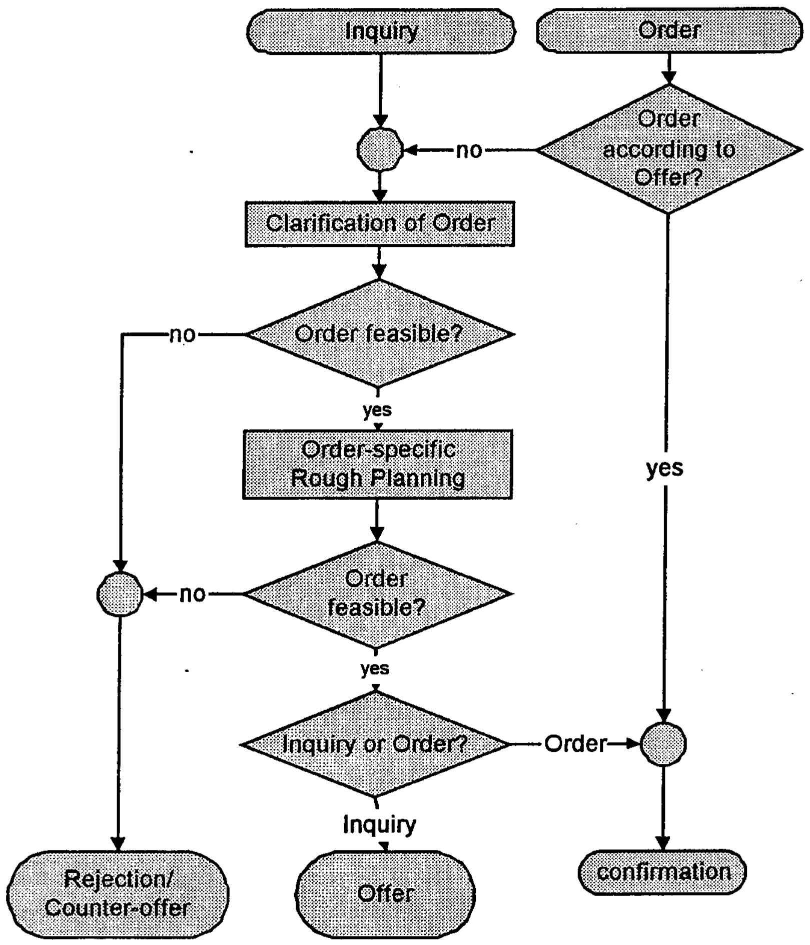


Figure 2: Process of proactive order coordination

Basic target figures, such as date and quantity targets from production planning, have to be backed up by continuous comparison of desired/actual values on a reconfirmation basis. Especially compliance of basic time limits is crucial in networks, due to the direct timewise interdependence of individual suborders.
Permanent supervision of order progress and early recognition of disruptions also fulfil the purpose of increasing both the transparency of order processing and the flexibility when reacting to internal and external discontinuities. However, the difference between internal and external disruptions is becoming less and less significant in production networks, since these enterprises unite to process orders together and thus have a shared interest in completing them properly.Causes for disruption are usually related to the uncertainties of production (including logistics) which lead to resource deficiencies, production and transport delays, and lower product quality.
In the following section a multi-agent-system for order specific rough planning as the most important function of order coordination in semiconductor industries will be presented, preceded by a short overview of the logistical problems in the semiconductor industry.

3. Overview of Semiconductor or Manufacturing

The semiconductor industry is characterised by an extremely turbulent and competitive environment where large numbers of different products and technologies have to be managed. The production of semiconductors takes place in four stages - wafer fabrication, wafer probe, assembly (packaging) and final testing. The physical location of the first two stages, and most commonly the final stage, is within one site, while the third stage and sometimes the final stage are integrated into another site. The production sites are scattered all over the world and often belong to legally independent companies. With joined forces they produce the final product, i.e. the chip:
The co-operation of independent businesses is certainly connected to a considerable coordination effort, further complicated by globally distributed production sites. Nevertheless, production networks must maintain or enhance their ability to respond quickly to changes and customer demands, otherwise they will not remain successful in today's competitive environment. The great innovative capabilities of the semiconductor industry lead to the rapid development of new products, which in turn not only shorten the product life cycle, but also quickly decrease the value of previous products. Time becomes a factor of increasing importance. Customer satisfaction is greatly enhanced by low prices and high product quality, short delivery times and delivery reliability (Warnecke, 1996). Due to the dynamism and the strong competition in the semiconductor industry, enterprises must be able to submit a binding offer in the shortest possible time. Accordingly, negotiations with customers are reduced to a few minutes[2]. At the same time, demands are growing regarding precision and the reliability of promised prices, delivery times, product quality and quantity. Neglecting customer demands brings about considerable competitive drawbacks (Richards, *et. al.*, 1997)

4. Concept and Architecture of the Multi-Agent-System

The focus of discussion regarding PPC-systems in transformable production networks is on organisation concepts which deal with the operation of production networks supported by a network co-ordinator[3]. The task of the network co-ordinator is both to optimise the supply chain to the advantage[4] of all network partners and to increase the efficiency of the network[5].
The functions of the network co-ordinator are not tied to any one person or institution. His/ her tasks can be adopted either by an enterprise which is in contact with customers, or by a neutral person. In some cases the customer can take on the tasks of the network co-ordinator.
The system presented here, is based on a software agent taking on the role of a network co-ordinator. The network co-ordinator co-ordinates the local plans of the network partners and tries to accomplish a globally optimal solution.
To include orders into the plan the network co-ordinator has to take into account the following facts:

- Network partners have their own interests.
- The tasks to be performed are dependent on each other.
- As a rule, each network unit has only incomplete information about all partners and the problem as a whole.

These aspects are also considered by electronic production agents which represent the network partners in the order co-ordination system. The task of the production agents is to offer their company's resources (manpower, stocks, equipment, capital, etc.) to the network co-ordinator, who pays for using them to carry out orders during a defined period of time. The production agents endeavour to make the highest possible profit from their available resources. This means that their ultimate goal is to obtain the greatest number of orders and maximum possible prices for their services. Three main features differentiate the production agents: the technologies available to them which allow for the

[2] Kurbel (1993) defines the ability to submit customised offers as quickly as possible as a competitive factor.

[3] Boucke et. al. (1996) are describing logistical scenarios where network-specific functions are performed by so-called network architects, co-ordinators or brokers. The network architect configures the production network, while the network co-ordinator is responsible for the operation of the production network, which includes the task of order coordination. The network broker negotiates with other networks, for example to tap external capacities.
[4] Cf. Wiendahl (1998): „Eine globale Optimierung des Netzes liegt (...) im Interesse aller."
[5] Cf. Hezel and Kulow (1998): „Empirischen Befunden zufolge bietet die Verknüpfung der gesamten Versorgungskette (...) ein enormes Effizienzsteigerungspotential."

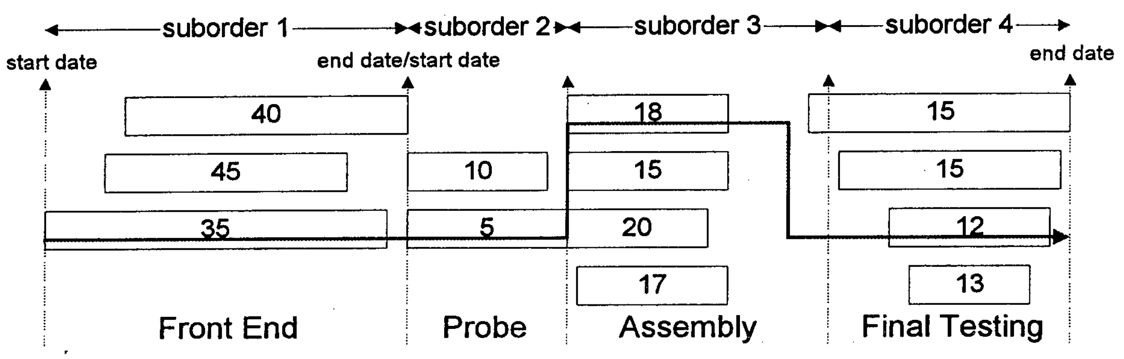


Figure 4: Selection of the best combination

transformation from one product into another; the production capacities available for the various technologies; and the cost of production. Technology is therefore crucial to what combinations of input and output are feasible and at what prices.

Each production agent has to submit offers for suborders at the request of the network co-ordinator. As basic time limits and production quantities are part of an offer, the production agent needs a rough planning function for its own production units.

A study done at the IPA showed that one possibility to accomplish the convergence of automated negotiations is to couple them with genetic algorithms. The following section describes briefly how they work.

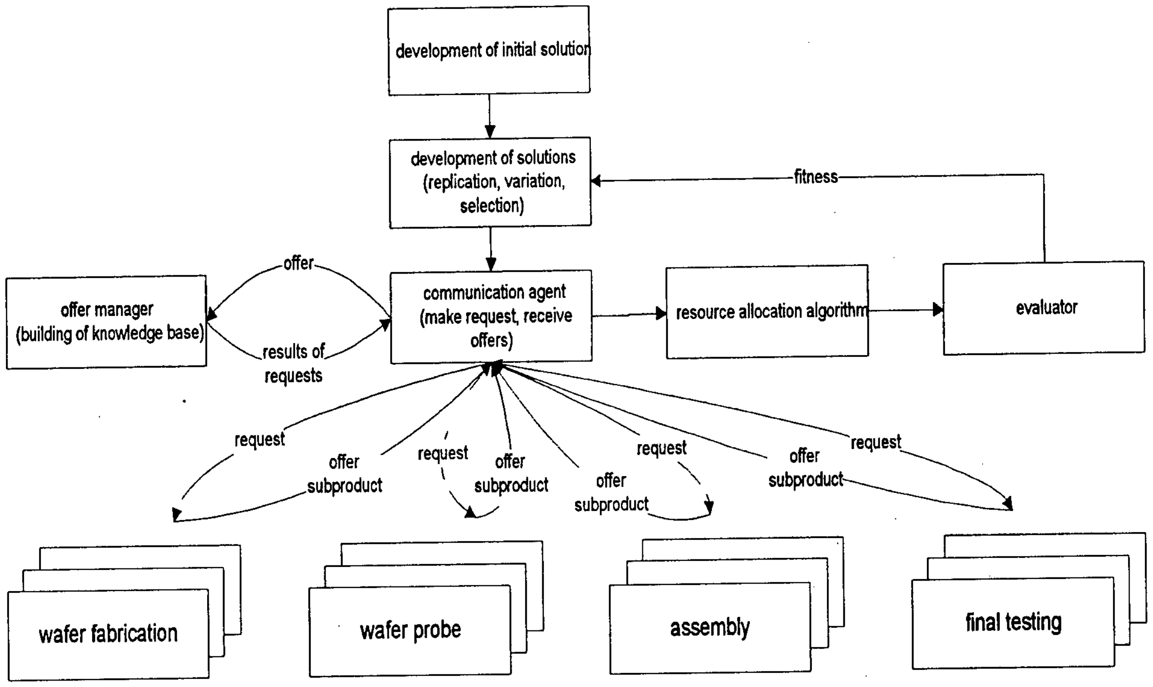
4.1 Genetic Algorithm

The goal of evolutionary algorithms and genetic algorithms is the 'breeding' of a good or improved solution to a given task (Weinberger *et al.*, 1994). The common basic structure all evolutionary algorithms share is the cyclic application of the evolutionary operations duplication (replication/ recombination), change (variation/mutation), and choice (selection) for a population of individual structures (Nissen, 1994). Each individual corresponds with a suggestion for a solution, encoded as vector form (chromosomes, genes) of an algorithm for the given task. The effects of evolutionary operations on the population depend on a suitability measure (quality, fitness) which evaluates the suitability of the individuals for the solution of problems. The suitability is usually determined not by the algorithm itself, but by an authority in its 'environment'. The cyclical application of evolutionary operations has the effect that the population of suggested solution gradually approximates a value (relative) of high or maximal suitability (Bäck *et al.*, 1995).

After initialisation and evaluation, the further evolution of the population of suggested solutions is cyclical up to the reaching of a specific criteria of breaking off (for example, obtaining a defined quality of the solution or the exceeding of a given number of generations or computer time). From the parents of the first generation, descendants are produced through the application of different duplication mechanisms. One distinguishing feature of this operation is that the suggested solutions of the descendants can exclusively contain information which can be acquired from the combination of different solution building blocks of the parents. On the other hand, changes in the descendants (mutation) cause new solution building blocks to enter the gene pool of the descendant generation.

Analogous to the evaluation of the start population, the descendants are evaluated. The selective operations based on this evaluation are responsible for choosing the parents for the following generation from the population of the current generation (usually including parents and descendants). The various mechanisms of selection tend to prefer individuals with high suitability. The selection of new parents completes the alternation in generations for the population.

Genetic algorithms are to be employed for optimisation, since they are suitable to solve multi-dimensional problems with non-linear or unsteady goal functions, as arise during production planning (Schulte, 1995). The degree of complexity of a problem is further increased by the examination of a manufacturing network as against single site production planning. Another advantage of genetic algorithms is their independence of the particular problem and their ability to be parallelised.



development of initial solution
development of solutions (replication, variation, selection)
fitness
offer
offer manager (building of knowledge base)
results of requests
communication agent (make request, receive offers)
resource allocation algorithm
evaluator
request
offer subproduct
request
offer subproduct
request
offer subproduct
offer subproduct
request
wafer fabrication
wafer probe
assembly
final testing

Figure 3: Combination of evolutionary algorithms and negotiations

Due to these characteristics, genetic algorithms are highly suited to control the automated negotiations within the order planning system.

4.2 Control of Automated Negotiations through Genetic Algorithms

Three tasks have to be solved for order rough planning. The order has to be divided into suborders according to the efficiency of the units. The division is executed according to the four steps of semiconductor production: wafer fabrication, wafer probe, assembly and final testing. The suborders have to be scheduled and assigned to the manufacturing sites. Genetic algorithms are employed for the scheduling of suborders. Based on valid initial solutions, new solutions are generated with the help of genetic operators (variation, mutation, selection). The solution is a vector shaped as follows:

(start date sub order 1, finish date sub order 1, ...start date sub order n, finish date sub order n)

Start dates and finish dates are positive integers. 0 represents today, 1 means tomorrow, 2 the day after tomorrow, etc. A consistency check guarantees that no inadmissible solutions are generated. An inadmissible solution is one that has a finish date of suborder n that is later than the start date of suborder n or suborder n+1.

For each vector of the population, the network co-ordinator sends an inquiry about the suborders including the corresponding frame schedule to the remaining network units (Fig. 3). The network partner (or the representing production agent) evaluates the inquiry with regards to price and feasibility (Fig. 3). By means of its rough capacity model and its ATP[6] function it can assess the feasibility of the request based on its knowledge of current operative capability, e.g. unused capacity, inventory, or lead time. If the order can be processed, the order promise agent makes an offer price while taking into consideration its enterprise's price policy and its own knowledge of the market.

The production agent which sends off the inquiries receives the offers, usually only for partial products or partial services, from the network partners. The resource allocation algorithm combines the partial offers of the individual partners into one offer for the complete product, the chip (Fig. 4). Then the combined offers are evaluated and the most favourable is selected. This way, the problem of assigning resources is solved

[6] A product is called Available-to-Promise (ATP) where an order can be filled from existing stocks. A product is Capable-to-Promise (CTP) where an order can be filled from existing stocks and is covered by the quantities calculated in the master production schedule.

as well. This means that for each solution of the population the best alternative is selected and the corresponding fitness function, described in detail in chapter 4.3, is calculated by the evaluator. Afterwards, the evaluated population is altered through genetic algorithms.

This method is repeated until the criterion for breaking off is reached. A break-off takes place if a previously determined time interval has expired or if a given quality of the solution is reached. This is the case if a solution is found in which the delivery date as desired by the customer is met and the costs for its realisation remain under a particular value which can be configured by the user.

Immediately after the optimisation through genetic algorithms the customer of the production networks receives one or more alternative offers. If the production network is awarded the order, the corresponding suborders are passed on to the individual units.

4.3 Goal System of the Production Network

It is the goal of the production network to manufacture the product requested by the customer at minimal cost while ensuring the desired delivery dates. The evaluation takes into account the prices demanded by the network partners for their services, transportation costs and inventory costs, as well as a parameter for the assurance of the desired delivery date.

The production network tries to get as many orders as possible and makes an effort to offer an acceptable price to the customer. In the same way, each partner tries to award the order for a partial service and, as a consequence, sets its own offer prices for suborders. The market mechanism prevents individual partners from asking for excessive prices. If several alternative partners are able to carry out one manufacturing sequence, the least expensive supplier has the best chance to win the order. If only one suitable unit exists, and if this unit tries to take advantage of this situation, this will lead to the disintegration or reconfiguration of the production network.

The price P_{intern} which has to be paid by all units participating in the processing of an order is:

$$P_{intern} = \sum_{l=1}^{no_{stage}} \sum_{Fab}^{VE} p_{l,Fab} \cdot \varphi \qquad (1)$$

with

$$j = \begin{cases} 1 & \text{Fab carries out production stage} \\ 0 & \text{otherwise} \end{cases},$$

no_{stage} = number of production stages, and

$p_{l,\,Fab}$ = price a fab asks for the execution of production stage 1

Those network partners which provide services are not able to include transportation costs into their offer, since during the negotiations it is still unclear which partners are going to receive suborders or which transport route the order involves. Thus, the price given in the offer never includes delivery. During the calculation of the total price, the evaluator generates the transportation costs $C_{transport}$ by means of a transport cost matrix.

$$C_{Transport} = \sum_{l}^{no_{stage}-1} \sum_{Fab_a}^{VE} \sum_{Fab_b}^{VE} c_{transport\ l,l+1} \cdot \alpha \cdot \beta \qquad (2)$$

with

$$\alpha = \begin{cases} 1 & Fab_a \text{ carries out production stage l} \\ 0 & \text{otherwise} \end{cases},$$

$$\beta = \begin{cases} 1 & FAB_b \text{ carries out production stage } 1+1 \\ 0 & \text{otherwise} \end{cases}$$

$c_{transport\ l,l+1}$ =transportation costs from manufacturing site 1 to 1+1

Inventory costs arise if after completion of one work cycle the following work cycle is not started immediately. The evaluation takes into account only those inventory costs which come up between different VE units. The inventory costs which arise during services of one network partner are included in the price setting of that unit. The inventory costs $C_{inventory}$ are calculated as follows:

$$C_{Inventory} = \left(\sum_{l=1}^{no_{stage}-1} (t_{start\,l+1} - t_{end\,l}) + t_{dd} - t_{no_{stage}} \right) \cdot \left(\sum_{i=1}^{l} \sum_{Fab}^{VE} p_{i,Fab} \cdot \varphi \right) \cdot \tau \qquad (3)$$

with

$$\varphi = \begin{cases} 1 & Fab \text{ carries out production stage} \\ 0 & \text{otherwise} \end{cases}$$

$t_{start\,l}$ = start date for production stage l

$t_{end\,l}$ = finish date for production stage l

t_{dd} = delivery date promised to final customer

The keeping of the desired delivery date cannot be measured directly in terms of a monetary quantity. Delivery reliability, however, contributes significantly to an enterprise's success in the long run. Non-compliance of delivery dates is a highly negative factor. In the fitness function applied here, the penalty function C_{delay} for non-compliance of delivery dates is user configurable. As a rule, the penalty function C_{delay} is defined as a function of delay and price. One obvious possibility is to assume a linear connection between C_{delay}. and the length of the delay.

$$C_{delay} = f(t_{dd} - t_{end\ stage}, P_{extern})$$
$$= \begin{cases} 0 & t_{dd} - t_{end\ operation} \geq 0 \\ \lambda \cdot (t_{end\ stage} - t_{dd}) \cdot P_{extern} & otherwise \end{cases} \quad (4)$$

for $\lambda \geq 0$ with

t_{dd} = delivery date promised to the customer

P_{extern} = price agreed upon with the customer

$t_{end\,stage}$ = date of completion of the last production stage

t_{dd} = agreed delivery due date

If contractual penalties were arranged with the customer, these can be used for the basis of the penalty function.

Following the above given considerations, the fitness function is:

$$F = P_{intern} + C_{transport} + C_{inventory} + C_{delay} \quad (5)$$

5. Practical Application

By means of an emulator the suitability of the procedure is verified and the modelling of the production network carried out. The emulator allows for the simulation of random configurations of production networks in the semiconductor industries. Like a real life network partner, the emulator is represented in the multi-agent-system by a production agent and has the task to simulate the behaviour of a network partner. In doing so, the emulator makes use of an event-based simulation model which replaces a concrete production unit. Each of the four production stages in semiconductor manufacturing is represented by its own simulation model mapped according to real-life conditions. The models share an interface with a database automatically supplied with relevant (historic) data (WIP, orders, work schedules, bills of material, etc.) of the corresponding production unit. The simultaneous use of several emulators enables the simulation of random production network configurations.
The models of production networks, which were used to check the full functionality and the behaviour of the coordination procedure, were generated with the help of realistic data for all production stages from three semiconductor companies.
Most recent tests have shown that through the global coordination of customer orders and production plans, which is accomplished by the system presented here, inventories at the network's interorganisational interfaces as well as the total processing times can be reduced significantly. Due to the high values of wafers and dies this way enormous costs can be saved.

6. Summary and Outlook

The presented method makes it possible to include orders into production nets' plans in real time. Simultaneously streamlining of the partners' production plans is accomplished. The distributed architecture and the algorithm based on automated negotiations react flexibly to all changes in the system. Thus, reconfigurations or local changes, such as the acquisition of new machines at one site, have no impact on the system.
Furthermore, a tool is now available which allows the production networks to schedule customer orders between different enterprises within a very short time. This is an important prerequisite for networks needed to present themselves as a cohesive unity to the customer. A reactive planning and control component which takes appropriate corrective action in case of disruption in networks is currently being developed. Through the co-operation of these two components the delivery

dependability will be improved considerably and customer satisfaction will be increased.
In general, the system can be transferred to production networks of other industries, if those networks consist of equal partners and if a suitable, clearly definable division of customer orders is possible.

Bibliography

1. Arnold, J., H.-M. Dudenhausen, H. Halmosi, H. (1996). Production Planning and Control within Supply Chains. In: Browne, J.; Haendler Mas, R.; Hlodversson, O. (Hrsg.): IT and Manufacturing Partnerships. Amsterdam: IOS Press, p. 139-149
2. Atherton, Linda and Atherton, Robert (1995). Wafer fabrication: factory performance and analysis. Kluwer Academic Publishers.
3. Bäck, T.; Beielstein, B.; Naujoks, B.; Heistermann, J., (1995). Evolutionary Algorithms for the Optimization of Simulation Models Using PVM. Arbeitspapier der Universität Dortmund, Fachbereich Informatik, Dortmund
4. Boucke, B. (1996). Entwicklung von Logistik-Szenarien. In: Dangelmaier (Hrsg.): Vision Logistik – Logistik wandelbarer Produktionsnetze zur Auflösung ökonomisch-ökologischer Zielkonflikte. Karlsruhe: Forschungszentrum Karlsruhe GmbH, 1996, S. 201-228
5. Dudenhausen, H.-M., Halmosi, H., Lickefett, M., (1996). Auftragsmanagement in Virtuellen Unternehmen, in: Industrie Management, Heft 6/96, S. 18-22
6. Hezel, H.; Kulow, B. (1998). Kooperation im Logistiknetzwerk – ein zukunftsweisender Weg zum effizienten Supply-Chain-Management. In: Auftrags- und Informationsmanagement in Produktionsnetzwerken – Konzepte und Erfahrungsberichte; 3. Stuttgarter PPS-Seminar F31; 18. Juni 1998; Stuttgart 1998, S. 53-78
7. Kurbel, K. (1993). Produktionsplanung und –steuerung : methodische Grundlagen von PPS-Systemen und Erweiterungen, München: Oldenbourg, 1993
8. Leachman, R. C., Benson, R. F., Liu, C. and Raar, D. J. (1996). IMPReSS: An Automated Production-Planning and Delivery-Quotation System at Harris Corporation - Semiconductor Sector Interfaces 26: 1 January-February.
9. Nissen, V. (1994). Evolutionäre Algorithmen: Darstellung, Beispiele, betriebswirtschaftliche Anwendungsmöglichkeiten. Dt. Universitäts-Verlag, Wiesbaden.
10. Sandholm, T.; Lesser, V (1995). Issues in automated negotiation and electronic commerce: Extending the contract net framework, in: Proc. First International Conference on Multiagent Systems (ICMAS -95), San Francisco
11. Schulte, J. (1995). Werkstattsteuerung mit genetischen Algorithmen und simulativer Bewertung. Dissertation Universität Stuttgart, Springer-Verlag, Berlin, New York.
12. Richards, H.D.; et. al. (1997): Flow of orders through a Virtual Enterprise: Their proactive planning & scheduling and reactive control. In: Computing & Control Engineering Journal, 1997, S. 73-79
13. Rosenschein, Jeffrey S; Zlotkin, Gilad (1994).
14. Massachusetts: MIT Press
15. Warnecke, H.-J. (1996). Effiziente Produktionsstrukturen und ihre Umsetzung in ein PPS-Konzept. In: wt-Produktion und Management 86 (1996), S. 532-536
16. Weinberger, T.; Keller, H.B.; Jakob, W.; große Osterhues, B. (1994). Modelle maschinellen Lernens. Symbolische und konnektionistische Ansätze. KfK-Bericht Nr. 5184, Kern-forschungszentrum Karlsruhe GmbH, Karlsruhe.
17. Westkämper, E.; Dudenhausen, H.-M. (1997). Optimization of order flow through virtual enterprises. In: Facilitating deployment of Information and Communication Technologies for competitive manufacturing / Fichtner, D; Mackay, R. (Hrsg.), 1997, S. 107-116
18. Wiendahl, H.-H. (1998). Zentralistische Planung in dezentralen Strukturen? Neue Organisationskonzepte versus alte Planungsmethoden. In: Auftrags- und Informationsmanagement in Produktionsnetzwerken – Konzepte und Erfahrungsberichte; 3. Stuttgarter PPS-Seminar F31; 18. Juni 1998; Stuttgart 1998, S. 79-108

COGNITIVE NETWORK – A NEW METAPHOR FOR MULTIAGENT PROGRAMMING

Galina V. Smerdina

Russian Research Institute for Artificial Intelligence,
Institute of Informatics Systems,
Lavrent'jeva Av 6, Novosibirsk, 630090, Russia
e-mail: galinas@iis.nsk.su

Abstract

A cognitive network is offered as a new model of knowledge representation. It is based on a uniform object platform that realizes principles of multiagent programming to describe problems with information given imprecisely. A cognitive network is a collection of intelligent objects connected by intelligent relations. The control over the network is carried out by a plan that is specified by a group of rules. A cognitive network represents knowledge in a hybrid way based on objects, constraints, rules and imperative programming.

Keywords: *Hybrid systems of knowledge representation, multiagent systems, cognitive network, knowledge systems methodology.*

1. Introduction

The history of the development of software engineering from algorithmic programming languages, frame-based and production systems up to systems of logic programming is an attempt to find adequate abstractions for mapping complicated systems and processes of the real world and optimal mechanisms of problem solutions. An object-based paradigm is a new step towards a proper reflection of the reality as any problem domain can be given with the help of objects and communication mechanisms. And at last the next stage of the development in this direction is multiagent environment where objects are endowed with intelligence.

The methodology of the system engineering of artificial intelligence assumes a choice of the most approaching mode of knowledge representation for a given problem or its part. It is obvious that we need construction of the programming environment for engineering of hybrid knowledge supporting a wide choice of alternatives of representation.

In this paper a new metaphor for multiagent applications that is a cognitive network, is offered. The specification of a problem domain as a cognitive network assumes the use of empirical rules of decomposition of the problem domain as the collections of intelligent agents connected by cognitive relations. The cognitive network integrates a hybrid knowledge representation on the basis of objects, relations, constraints, subdefinitness, rules, and imperative programming. The examples introduced in a paper show that such representation expands the range of application of multiagent systems.

2. From a semantic network to a cognitive one

The idea of cognitive network has arisen from an experience of work with knowledge representation on the basis of the semantic network in the system Semp-F [1,2,3]. This experience has shown a possibility of its modification and adaptation to the problems with cooperative agents. A base structure in Semp-F is a semantic network that represents a current set of active objects and relations. Active objects are characterized by their attributes and constraints given as functional dependencies of attributes. Constraints link attributes of the same object and the attributes of different objects as well. A network access is carried out through production rules, in the left part of which the operation of pattern matching is used. The semantic network in Semp-F maps an information extracted from natural language texts, and for user the traditional mediums of work with a semantic network (pattern matching) are submitted.

In practice we more often meet with the representation of à structured problem domain and very successful for this purpose the map it as set of cooperative agents is. Cooperation is ability of an agent to work with other agents to perform a task and as such is a fundamental property of an agent. An agent in contrast to the active object in Semp-F has property of autonomy and reactivity. The property of autonomy assumes symbolical description of a fact with the help of the unique gang of attributes and constraints on them. Reactivity is ability of an agent to react fast without the use of complicated reasoning. Along with properties defining as set of attributes of object and constraints on them, agent has some behavior depending on exterior or interior conditions. The behavior of agent is described with the help of rules, in the left part of which a condition on attributes of agent is set. As the agent a relation that connects two or several objects can appear. The agent-relation or the cognitive relation describes cooperating knowledge of agents and contains symbolic model of the outside world about which agents develop plans and make decisions. The agent-relation can have a property of reactivity as well. The cognitive network is a network that represents the current set of agents or more precisely the collection of agents-objects connected by agents-relations. Such representation allows us to combine reactive and cognitive approaches for the construction of full solution in real applications. The mechanism of constraints based on subdefinite computing model [4,5] helps dynamically to adapt the behavior of agents to modifications of external environment. The specification of constraints on the basis of subdefinite computing model represents the natural mode of description of restrictions.

The suggested model is the development of the model offered in [6]. It takes into account modern developments in area multiagent systems [7], process engineering of constraints programming [4,5] and various modes of the communications of agents: cognitive, reactive or hybrid [8].

3. Basic concepts of cognitive network

3.1. Cognitive network (CN)

Cognitive network is a collection of intelligent agents-objects connected by intelligent agents-relations. An agent-object in cognitive network has the properties of autonomy and reactivity. The agent-relation has the property of reactivity as well and has access to attributes of linked agents-objects. Agents are described with the help of classes by using the mechanism of inherithing.

3.2. Agent-object (O)

The knowledge base of the agent-object can be represented by the triple:

$O = <A, C, R>$,

where $A = \{a_1,.., a_k\}$ – attributes described by a name and a type. For attributes the following types can be defined: *integer, real, atom, subdefinite integer*, given by the values as an interval, for example: *start: integer (10.. 20)*. Except simple types for description of agents the structured types of data such as a set and a tuple can be used.

This part creates dynamic components of agent knowledge and maps symbolical representation of facts both about the exterior world and system interior conditions.

$C = F(A)$ – functional dependencies that specify constraints on attributes, for example, *finish = start + duration*. Constraints represent the static part of an agent knowledge.

$R = \{r_1,.., r_k\}$ – serially ordered set of rules as r_i: *Cond -> Act*,

where *Cond* – conditions on attributes of the agent, *Act* – evaluations over attributes or operations on output of the values of attributes. Each rule is executed once, if an operator *repeat* specially is not used that calls cycling the rule.

This part contains knowledge about behavior of the agent.

3.3. Agent-relation (*L*)

The agent-relation $L(O_1,.., O_n)$ can link 2 or more agents-objects. It allows us to create functional dependencies on attributes of various agents-objects one or different classes and rules of interaction of agents-objects as well.

The knowledge base of the agent-relation can be represented by the triple:

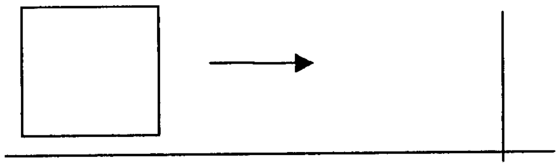
$L = <A, C, R>$,

where $A = \{a_1,.., a_k\}$ – attributes of the cognitive relation. These attributes are set similarly to the agent-object by a name and a type.

$C = F(a_1,..,a_k \in O_1,.. ,a_1,..,a_k \in O_n)$ – functional dependencies on attributes of agents-objects connected by this relation.

$R = \{r_1,.., r_k\}$ – serially ordered set of rules as r_i: *Cond -> Act*,

where *Cond* – conditions on attributes of objects connected by this relation, *Act* – evaluations over attributes of the relation and over attributes of linked agents-objects or output of values of attributes. Each rule is executed once, if an operator *repeat* specially is not used that calls cycling a rule. The operator *activate* used in *Act* allows us to transmit control to other agents-objects or agents-relations. This part can be used for the representation of direct inference.
For the agents we distinguish 2 such as the communications: output of information about specified values of attributes and refinement or modification of controlled attributes. The example of the description of the agent on the MARS language [9]:
The agent represents a square located in a parallel way to axis. The square independently moves to the right, while will not come across yet a motionless wall.



```
  structure POINT
   x: integer (0.. 100);
   y: integer (0.. 100);
 end;
 class ACTIVE _ SQUARE
    centre: POINT;
    side: integer (10.. 20);
    state: (move, stop);
 constraints
   centre.x <  75;
 reaction
   state = move = > centre.x: = centre.x + 1;
   Vis _ square (centre, side); repeat;
               // on repeated  execution of a rule
   = > state: = stop;
 end;
```

In this example the *ACTIVE _ SQUARE* class is circumscribed, for which the *centre* and *side* attributes are defined. In a section ***constraints*** a restriction is given that inspects the transition of the square. In a section ***reaction*** 2 rules are given that determine behavior of the square.

4. Control by the cognitive network

The evaluations are produced when the current set of agents-objects and agents-relations is created, that is when the cognitive network is generated. The control model contains the sequence of rules, that is a plan. To control the activation of agents in a network an operator *activate* is used.
The evaluation in cognitive network is produced in the following order:
1. The functional network is generated that links attributes of agents indicated in *constraints*, then evaluations in the network are produced, therefore some attributes can receive new more precise values. The evaluations happen on a data-flow principle, that is the agents exhibit the properties of activity to attributes: when the modification of value of one attribute arises the values of other attributes connected with him by functional dependencies vary. The functional network puts into practice subdefinite computing model that theoretical background in works [4, 5] is given.
2. The agents-objects or agents-relations indicated in the operator *activate* are activated.
3. For the activated agent the rules from a section *reaction* are executed sequentially. If in the rule the operator *repeat* is used then the rule is executed so long as in the functional network there will be no inconsistency. In this case it happens rollback on the previous step of evaluations and the next rule is performed.
The library of plans can be created that contains the set partially of developed plans, each of which contacts with a condition of call.
For a cognitive network the following operations are defined:

- *Create* –to create a current set of agents,
- *Add* – to add a new agent,
- *Delete* – to delete an agent from a network,
- *Edit* – to edit attributes for a given agent.

An example of the creation of network and control by network is:

```
 create Active _ square: ACTIVE_ SQUARE (side
= 50, centre = 0,0);
  plan
    read (cl) = Enter = >
    Active _ square.state: = move;
    activate (Active _ square);
  end;
```

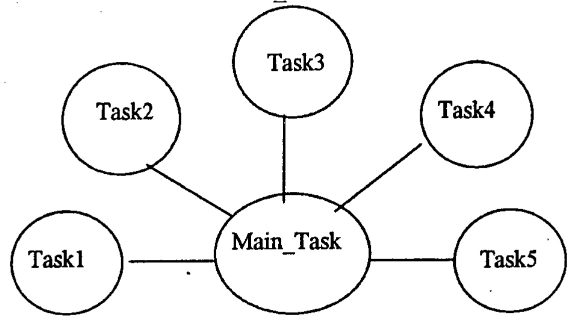
The current set of agents is created by an operator *create* and the one agent *Active _ square* of the class of agents *ACTIVE_SQUARE* contains. Plan determines the control model and contains one rule that activates the square with key *Enter*. The first rule in *reaction* is executed until inconsistencies appear, that is while the square does not reach the wall. When inconsistency takes place the next rule is executed.

5. Demonstration of a cognitive network on examples

Shown below 3 examples demonstrate the use of the offered model for the solution of the different tasks. In the first example the cognitive network is used primarily as demonstrating of the subdefinite model of evaluations with inexact given information and represents the static model of interrelationship of agents. The second example is characteristic for traditional multiagent interaction and describes cooperative behavior of agents in time. Third one demonstrates the possibility of inference by applying a sampling strategy. The given examples can seem trivial, but behind it there is simplicity and clearness of the universal use of cognitive network for the tasks traditionally divided by researchers.
The program on the MARS language represents the description of class of agents-objects and agents-relations, the creation of the current set of agents, i.e. cognitive network and plan that specifies model of control.

5.1. Example 1

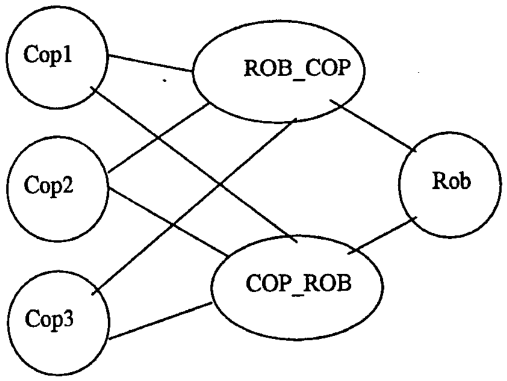
A task consists of 5 subtasks, it is required to define maximum time necessary for realization of the task. It is known that the subtasks 2,3 and 4 should begin after the termination of the subtask number 1, the subtask 4 should begin after the realization of the subtask 2, and subtask 5 – after the termination of the subtasks 3 and 4. Besides it is known that the duration of the realization of the tasks is inexact and is set as intervals. In this example the duration of the task is defined in conditional units. It is required to estimate the execution time of the task in whole.
The cognitive network consists from 5 of the agents-objects of the class *TASK* and one agent-relation of the class *MAIN _ TASK*:



```
  type TASKS (Task1, Task2, Task3, Task4,
Task5);
  class TASK
    name: TASKS;
    start: integer;
    finish: integer;
    duration: integer;
  constraints
    finish = start + duration;
  end;
     relation MAIN _ TASK (task1, task2, task3,
                task4, task5: TASK);
    start: integer;
    finish: integer;
  constraints
    task1.start = start;
    task2.start > = task1.finish;
    task3.start > = task1.finish;
    task4.start > = task2.finish;
    task4.start > = task1.finish;
    task5.start > = task4.finish;
    task5.start > = task3.finish;
    finish > = task5.finish;
  reaction
    = > print (" time necessary for realization of
the task equal ", finish);
  end;
  create Task1: TASK (duration = (10.. 12));
          Task2: TASK (duration (5.. 30));
          Task3: TASK (duration (10.. 20));
          Task4: TASK (duration (3.. 40));
          Task5: TASK (duration (10.. 20));
      Main _ Task: MAIN _ TASK (Task1, Task2,
Task3,Task3,Task4,Task5:TASK);
  plan
    => activate (Main _ Task (start: = 0));
  end;
```

5.2.Example 2

The next example is adapted from [10] and serves to illustrate a pursuit game. In this domain, there are four agents, one robber and three cops that are placed in a rectangular grid limited by *Wx* horizontal and *Wy* vertical units. Agents can move in four directions: up, down, left, and right. The goal of the game is to capture the robber that can be done only at a wall or in a corner. Each of the five agents can stay in the current position or move by one step in the above four directions. The robber and the cops cannot move simultaneously: in each move, either the robber is moved or all the cops at once. The individual plan for cops consists of one action only: if the distance to the robber is greater than one, move so as to decrease the distance by 1. Two agents' plans are in conflict, if the agents want to move to the same square. The cops can coordinate their actions. In this example we do not attempt to give an efficient procedure for capturing the robber; rather, we only want to demonstrate the method of knowledge representation. The model will be shown as a cognitive network containing three cop agents, one robber agent, and two relations: between the robber and the cops, and between the cops and the robber.



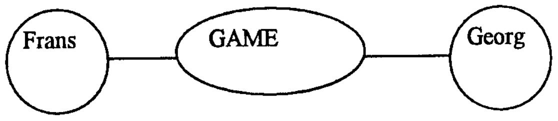
```
class ROBBER
  x : integer (0..10);
  y : integer (0..10);
end;
class COP
  x: integer (0..10);
  y: integer (0.,10);
end;

relation COP_ROB (cop1: COP, cop2: COP,
cop3: COP rob: ROBBER)


constraints
  cop1.x != cop2.x != cop3.x;
  cop1.y != cop2.y != cop3.y;
reaction
  rob.x – cop1.x > 1 => cop1.x := cop1.x + 1;
  cop1.x – rob.x > 1 => cop1.x := cop1.x – 1;
  rob.y – cop1.y > 1 => cop1.y := cop1.y + 1;
  cop1.y – rob.y > 1 => cop1.y := cop1.y – 1;
rob.x – cop2.x > 1 => cop2.x := cop2x + 1;
  cop2.x–rob.x > 1 => cop2.x := cop2.x – 1;
  rob.y - cop2.y > 1 => cop2.y := cop2.y + 1;
  cop2.y – rob.y > 1 => cop2.y := cop2.y – 1;
rob.x – cop3.x > 1 => cop3.x := cop3.x + 1;
  cop3.x – rob.x > 1 => cop3.x := cop3.x – 1;
  rob.y – cop3.y > 1 => cop3.y := cop3.y + 1;
  cop3.y– rob.y > 1 => cop3.y := cop3.y – 1;
end;
relation ROB_COP (rob: ROBBER, cop1: COP,
cop2: COP, cop3: COP)
constraints
  rob.x != Wx;
  rob.y != Wy;
reaction
  (rob.x – cop1.x) > 0 &
  (rob.x – cop2.x) > 0 &
  (rob.x – cop3.x) > 0 => rob.x := rob.x + 1;
  (rob.y - cop1.y) > 0 &
  (rob.y – cop2.y) > 0 &
  (rob.y – cop3.y) > 0 => rob.y := rob.y + 1;
  rob.x = Wx & rob.y = Wy =>
  print ("give up");
end;
create rob: ROBBER (x = 5, y =5);
      cop1:COP (x = 1, y = 4);
      cop2:COP (x = 4, y = 1);
      cop3:COP (x = 2, y = 2);
     ROB_COP (rob, cop1, cop2, cop3);
     COP_ROB (cop1, cop2, cop3,rob);
plan
  => activate (COP_ROB, ROB_COP); repeat;
end;
```

In this example the use of n-arity relations, the specification of constraints and coordinated behavior with the aid of these relations is illustrated.

5.3. Example 3

The logical-arithmetical puzzle, it is adapted from [11]. Fransua and Georg play in tennis. Fransua wins Georg with the score six three. Fransua loses one feeding, and Georg – three, the last time wins Fransua. Who had the first feeding?

The cognitive network contains two agents-objects, i.e. two players: Fransua and Georg and one agent-relation GAME that is characterized by the set of feedings and by the set of prizes and the rules of game describes.



```
type Player (Fransua, Georg);
type Prize INTEGER (0,1); // 1 - prize, 0 - loss
  type Feeding INTEGER (0,1); // 1 - feeding, 0 -
           feeding at the contender
class PLAYER
  name: player;
```

```
    service: tuple of Feeding [9]; // only there were
                    9 feedings
    gain: tuple of Prize [9]:
    total _ gain: integer;
  constraints
    total _ gain = sum item in gain: item; // only of
                    won feedings
  reaction
     service [0] = 1 = > print (" by first given ",
                    name);
  end;
    relation GAME (player1: PLAYER, player2:
                    PLAYER)
  constraints
         // Gives Fransua or Georg
    forall item in player1.service:
  player1.service [item] = 1 - player2.service
[item];
         // Fransua and Georg give on queue
    forall item in player2.service:
    player2.service [item] = 1 -player1.service
                    [item-1];
         // Benefits Fransua or Georg
    forall item in (player1.gain:
  player1.gain [item] = 1 - player2.gain [item];

         // The last time benefits Fransua
  player1.gain [9] = 1;
       // constraints on lost feedings
      (sum item in 0.. 8: (player1.service [item] *
              (player2.gain [item]) = 1;
       (sum item in 0.. 8 (player2.service [item] *
              (player1.gain [item]) = 3;
  end;

  create
    Frans: PLAYER (name: Fransua, total _ gain:
                    6);
     Georg: PLAYER (name: Georg, total _ gain:
                    3);
    Game (player1: Frans, player2: Georg);
 plan
 = >
 try
    edit Frans (service [0]: 1, gain [0]: 0)
          edit Georg (gain [1]: 0, gain [3]: 0,
                 gain [5]: 0)
 or
      edit Georg (service [0]: 1, gain [0]: 0, gain
                    [2]: 0);
    edit Frans (gain [1]: 0);
 end;
  activate (PLAYER);
      end;
```

The inference is based on check of inconsistencies in the network by applying sampling strategy: the supposition at first is made that first was given Fransua and then Georg The computing model is initialized by values in the supposition that the order of the won games is not important and only their number is taken into account.

6. Discussion

Suggested model of knowledge representation is not a simple expansion of semantic network. Combination of well-known means of knowledge representation such as objects, constraints, productions that represent behavioral cognitive pattern or simple reactions along with an expansible set of data types lead to a new quality, therefore one can say about a new model of knowledge representation, i.e. cognitive network. Such model represents a reality as it is through a population of certain entities that exist in a certain restricted environment and interoperate according to well-known lows and, if needed, can perform coordinated actions. Generally accepted understanding of an agent as a system whose behavior is goal-oriented toward a certain state of the world, is expanded up to system that reflects a certain entity imbedded into a context of the fragment of external world. And interdependencies between these entities with the help of cognitive relations are described. Such approach allows us to see obvious connection with MAS organizations, MAS interaction protocols, theories of dependencies.

7. Conclusion

The cognitive network is a new paradigm that offers us to consider the problem domain as a collection of agents-objects and agents-relations. Such decomposition satisfies many problems for representation, inference and refinement of knowledge assigned in symbolical mode with incomplete and inexact initial data.

The properties of the cognitive network are:

1. Activity. The property of activity consists in the fact that the network is capable to calculate itself. Modification of the values of one attribute calls automatic reevaluation of other attributes connected by constraints.

*2. **Reactivity***. The property of reactivity consists in the fact that cognitive network is capable to make decision depending on the state network or to change a state of the agents by calling a runtime check behind restrictions.

3. ***Adaptive control.*** The special appeal of the offered model consists in a possibility of dynamic adaptive control by cognitive network at the expense of dynamic modification (adding a new agent or deleting old one) and modification of conditions of a plan, for example, with the help of events.

The offered model of knowledge representation, the cognitive network, reflects wider understanding of multiagent interaction than conventional, taking into account both static and dynamic models of interaction of agents, and thus expands the range of the tasks considered from the point of view of multiagent programming.

This model can be applied in different knowledge domain: from real time planning, scheduling, and resource allocation up to the expert systems and problems of adaptive control in conditions of varying environment, for example, an automatic control of the automobiles on a given route.

As a toolkit for construction of the cognitive network, the system MARS-V is realized using the modern facilities of knowledge representation with the help of agents and visual map of modelling process.

The system MARS-V is open environment for incremental development of applications, the base variant of which assumes the use of the C++ compiler, C++ libraries, libraries of subdefinite data types, set of program components for visual description of the structure of problem domain, and specification of knowledge components.

Bibliography

1□ Yu. A. Zagorulko, I.G. Popov. A Software Environment based on an Integrated Knowledge Representation Model // Perspectives of System Informatics (Proc. Of Andrei Ershov Second International Conference PSI'96). - Novosibirsk, June 25-28, 1996. -P.300-304.

2□ Yu. A. Zagorulko, I.G. Popov. Knowledge Representation Language with Objects and Constraints // Proc. Of 6th East-West International Conference on Human-Computer Interaction - Human Aspects of Business Computing (EWHCI'96). -Moscow, Russia, 12-16 August, 1996. -P.56-66.

3□ Yu.A. Zagorulko, I.G. Popov. Object-Oriented Language for Knowledge Representation Using Dynamic Set of Constraints // Proc. Of the Third Joint Conference on Knowledge-Based Software Engineering in Smolenice, Slovakia, 1998, -P.124-131.

4□ Vitaly Telerman, Dmitry Ushakov. Data Types in Subdefinite Models. In: Jacques Calmet and others (eds)., Art. Intell. And Symbolic Mathematical Computation, Lecture Notes in Computer Science; Vol. 1138, Springer, (1996). -P.305-319.

5□ Igor Shvetsov, Vitaly Telerman, Dmitry Ushakov. NeMo +: Object-Oriented Constraint Programming Environment Based on Subdefinite Models. In: Gert Smolka (eds)., Principles and Practice of Constraint Programming - CP97, Lecture Notes in Computer Science; Vol. 1330, Springer, (1997). -P.534-548.

6□ Christoph Welsch, Gerhard Barth. Reasoning Objects with Dynamic Knowledge Bases. Lecture Notes in Artificial Intelligence. // Springer-Verlag. September 1989. N390. -P.257-268.

7□ Michael Wooldridge, Nicolas R. Jennings. Agent Theories, Architectures, and Languages: A Survey. Lecture Notes in Artificial Intelligence. // Springer-Verlag. August 1994. N890.-P. 1-39.

8□ Z. Guessoum, M.Dojat. A Real Time Agent Model in an Asynchronous-Object Environment. Lecture Notes in Artificial Intelligence. // Springer-Verlag. January 22-25. 1996. N1038. -P.190-203.

9□ G.Smerdina. MARS: Multiagent Active-Reactive System. Sixth National Conference with International Participation "CAI`98". -Pushchino. Russia, 5-11 October. -P. 59-65.(In Russian).

10□ F. Von Martial. Coordinating Plans of Autonomous Agents. Lecture Notes in Artificial Intelligence. // Springer-Verlag. January 1992. N 610. -P 109-113.

11□J.L. Laurier. Artificial Intelligence Systems. Translation by V. Stefanuk. -Moscow, Russia, 1991, -P.485-487.

AGENT-BASED KNOWLEDGE MANAGEMENT FOR CONCURRENT ENTERPRISE CONFIGURING

Alexander V.Smirnov

St.Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS),
SPIIRAS, 39, 14th Line, St.Petersburg, 199178, Russia
e-mail: smir@mail.iias.spb.su

Abstract

The fast development of transportation and communication means lead to emerging global businesses and such a new form of co-operation as Concurrent Enterprises (CE). Interest in CE and research supporting such industrial groups is growing along with increasing use of AI-based engineering and management technologies, and the trend towards product data and knowledge globalisation. Based on CE concept a product should be designed for CE, as much as the CE should be designed for the product. CE configuring is a new approach to industrial network enterprise creation and reuse that considers enterprises as assemblies of reusable components (units) defined on a common domain knowledge model "product – process - resources". The goal of Knowledge Management (KM) is to facilitate knowledge transfer and sharing of in the context of CE structures integrating the customer and the supplier. KM tools for CE configuring activate reusable components and configuration knowledge as needed to interactively assist users (agents) while creating new CE configurations and new reusable components The paper discusses a generic methodology and an agent-based environment architecture for CE configuring knowledge management.

Keywords: *concurrent enterprises, configuration, constraint network, agent, decision making*

1. Introduction

Lately a new development trend in domain of electronic business collaboration is observed. Applying Concurrent Engineering and Electronic Commerce in the context of inter-enterprises business collaboration within the Virtual Enterprise is named Concurrent Enterprising (CE-NET, 1998).

The globalisation of industry has increased the need for industry standardisation as a methodology to avoid duplication and misinterpretation. Knowledge as the critical resource for business activity characterises this era. Every company has its own intellectual capital. Capitalising on corporate collective knowledge and intellectual assets takes more than an Intranet, some search and document management facilities. Today corporate knowledge of large industrial companies is distributed among many DBs. For example, engineering and manufacturing DBs that contain data about the product design and production, domain DBs that contain information about product use at the end-user sites and problem-oriented applications, financial DBs, and marketing DBs, etc.. Need to manage industrial knowledge to convert external market forces (such as speed of change, cycle-time reduction, globalisation, etc.) and certain internal changes. In result new information technologies like Agent Technology, Knowledge Management and other are attracting increasing interest from industrial companies.

Starting from the evaluation of existing enterprise integration architectures (CIMOSA, GRAI/GIM and PERA), the IFAC/IFIP Task Force on Architectures for Enterprise Integration has developed an overall definition of a generalized architecture. The proposed framework was entitled GERAM - Generalized Enterprise Reference Architecture and Methodology (ISO TC 184/SC 5/WG 1, 1997).

GERAM considers those methods, models and tools, which are needed to develop, design, build and maintain the integrated enterprise, be it a part of another enterprise, a stand-alone enterprise or a network of enterprises (virtual enterprise or extended enterprise). GERAM defines a tool-kit of concept for designing and maintaining enterprises for their entire life history. GERAM is not just another-proposal for a new enterprise reference architecture, but is *meant to organize existing enterprise integration knowledge.*

The goal of an enterprise model is to achieve model-driven enterprise design, analysis, and evaluation. In (Gruninger, 1997) an enterprise organization ontology is considered as a set of constraints on the agents' activities. Currently multi-agent systems and intelligent agent technologies are widely used for CE applications (Beyer *et al.*, 1999; Bussmann, 1998; Fischer *et al.*, 1996; Jennings, 1996; Pawlak *et al.*, 1997; Sandholm, 1998; Scherer and Katranuschkov, 1999).

General objectives of this paper are to develop a generic methodology and an agent-based environment for CE configuring knowledge management

2. CE configuring

Product reconfigurability is becoming more and more important. Reconfiguration is necessary when the customer wants either to upgrade the product to include new or better functionality, or to replace nonfunctioning parts for which identical replacements no longer exist. Instead of exchanging the whole product, the producer can add or replace certain parts to provide the desired functionality. Reconfiguration is a complex task, because replacing one component might affect other components, which in turn would have to be replaced or adjusted, and so on. The reconfiguration basically performs the process of adapting old configurations to new situations.

In order to design a CE that can be reconfigured to meet the changing production demand, one has to understand the relationship between the structure of the system "product - business process - CE resources". CEs are able to trade product models on Business Network that make possible the truly integrated virtual supply chain. CE configuration generates customised solutions based on a standard components (as templates or baselines) or CE model. Configuration consists of two aspects – configuring/reconfiguring and configuration maintenance. Configuring deals with creating configuration solutions; it involves selecting components and the ways of their configuring. Configuration maintenance deals with maintaining a consistent configuration under change; this requires the consistency among the selected components and decisions. When a decision for selected components changes, configuration maintenance must trace all the decisions related to the changed decision and revise them, if necessary, to maintain consistency among the components and decisions.

Configuration is an application area of AI that deals with assembly of complex system composed from a set of simpler components. Various approaches are implemented in the field of configuration (Stumptner, 1997): (i) representation-oriented approaches (rule-based configuration, structure-based configuration, constraints satisfaction problem, dynamic constraints satisfaction problem, resource-based configuration, component-oriented configuration, constructive problem solving); (ii) task-oriented approaches, (iii) case-based reasoning, (iv) hybrid systems.

CE configuring consists of two stages: CE modeling and CE maintenance. CE maintenance requires to use a special technology called Corporate Knowledge Management (KM). KM is a term that covers processes and technologies to store and exploit the know-how and experience of company employees. It comprises acquisition, analysis, transformation, storage and dissemination of knowledge throughout the complete engineering and management (E&M) processes.

3. CE knowledge management

An important requirement for collaborative system is the ability to capture knowledge from multiple domain and store it in a form that facilitates re-use and sharing (Bradshaw, 1993; Neches, 1991; Patil, 1991). KM could be identified by four factors behind successful KM systems (Donkin, 1998): an understanding by employees why knowledge sharing is important, recognition by employees, the legacy of existing practices, and a support mechanism or safety net, which allows employees to experiment. KM is 90 per cent people and 10 per cent technologies, in result the four general functions of KM - Externalization, Internalization, Intermediation and Cognitive (Delphigroup, 1998) describe the relation "user – knowledge/data bases".

A KM system addresses all three elements of a knowledge process (Livelink, 1998):

- Knowledge discovery – people search for any and all information that can help them to get their jobs done and achieve their organization business objectives.
- Knowledge Organization – the collection and management of information of all types – from files, documents and objects to running histories of enterprise projects, the contributors, the consensus, the solutions, and the activities.
- Collaborative Knowledge Development – people taking action and working together to create project, form teams, develop project deliverables, and manage projects and processes through their many stages and cycles.

4. CE model

Applying GERAM approach enables to form the CE conceptual model of the system "product - process - resource" satisfying the constraints on manufacture resources. The CE configuring stage is represented by relation "configuring of the product (product structure, materials bill) -> configuring of the business process (process structure, operation types) -> configuring of the resource (structure of system, equipment and staff types)".

The conceptual model of product is hierarchical. Such model as STEP advocates a hierarchical approach to product description. The STEP model divides the product into four levels: product, product version, subassembly, part. The conceptual models of manufacturing systems are also hierarchical. Such models as CAMT-I and NIST advocate a hierarchical approach to manufacturing system design and analysis (Nadoli, 1993). The CAM-I model divides the factory management system into four levels: factory, job shop, work centre, and resource. The NIST model considers five levels: facility, shop, cell, workstation, and equipment.

The implementation of CE approach is based on the shareable information environment that supports the "product - process - resource" model (PPR-model) used for integration and co-ordination of users activity. This model is studied from different aspects – viewpoints or users group (Figure 1). CE model as PPR-model could consist of three parts: CE Product Model, CE Process Model, CE Resource Model. Figure 1 describes links of users and knowledge&data with CE Model parts for general business processes from supply chain.

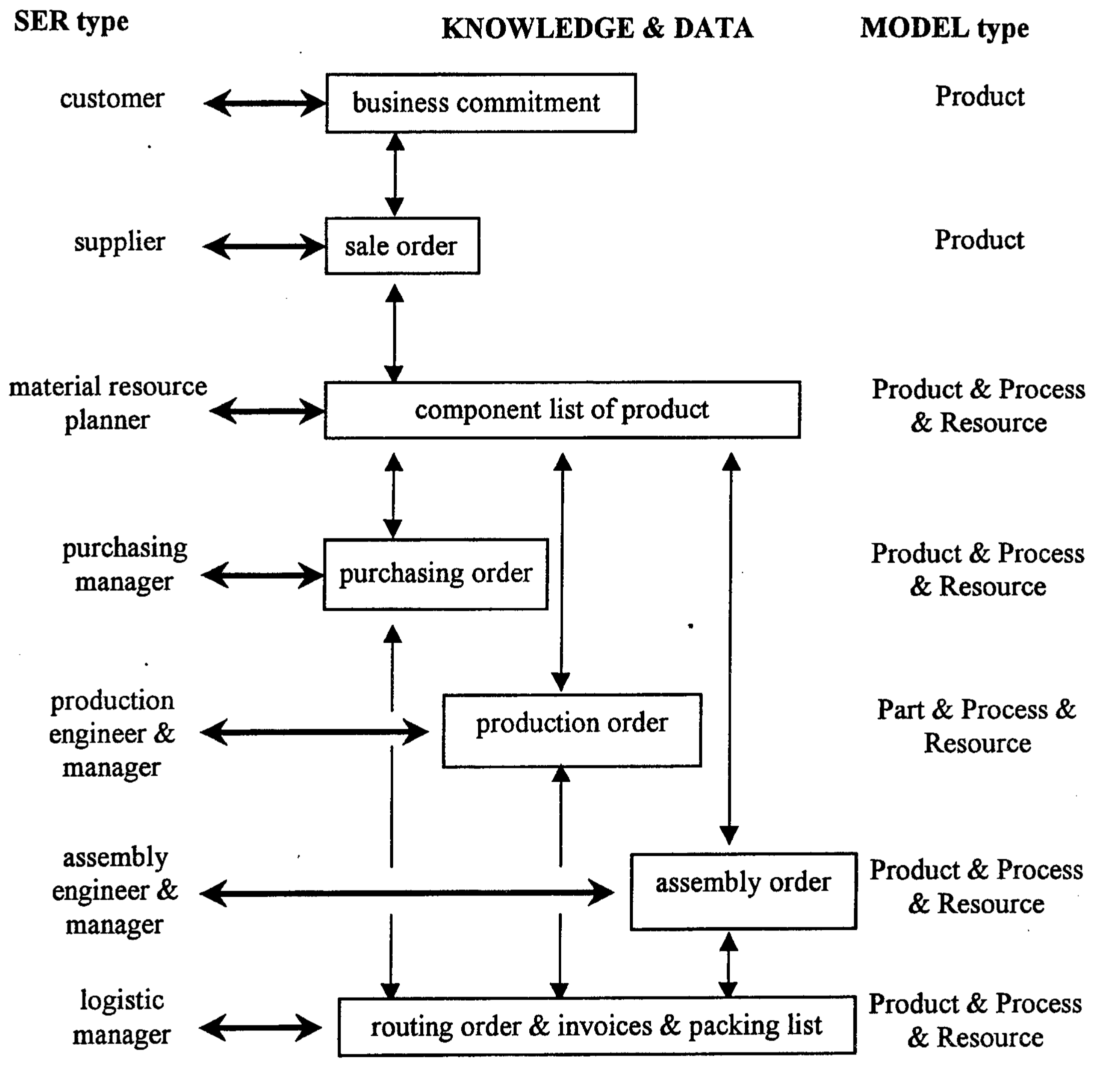

SER type
KNOWLEDGE & DATA
MODEL type
customer
business commitment
Product
supplier
sale order
Product
material resource planner
component list of product
Product & Process & Resource
purchasing manager
purchasing order
Product & Process & Resource
production engineer & manager
production order
Part & Process & Resource
assembly engineer & manager
assembly order
Product & Process & Resource
logistic manager
routing order & invoices & packing list
Product & Process & Resource

Figure 1. Links of users and knowledge&data in CE (example)

Enterprise Modelling Languages as a part of GERAM define the generic modelling constructs for enterprise modelling adapted to the needs of people creating and using enterprise models. In particular enterprise modelling languages will provide constructs to describe and model human roles, operational processes and their functional contents as well as the supporting information, office and production technologies. Examples of modelling languages can be found in ARIS, CIMOSA, GRAI/GIM, IEM or the IDEF family of languages. Relevant Standardization:

- CEN ENV 12 204 "Constructs for Enterprise Modelling" defines a reference set of twelve constructs for enterprise modelling (business-process, capability set, enterprise activity, enterprise object, event, object view, object state, order, organizational unit, product, resource, relation).
- ISO DIS 14258 defines rules and guidelines for enterprise models.

Enterprise models may be defined in various ways. In increasing order of formality generic enterprise models may be defined as (ISO TC 184/SC 5/WG 1, 1997):

- Natural language explanation of the meaning of modelling concepts (*glossaries of terms*).
- Some forms of meta models (for example, *entity relationship meta schemas conceptual models of· terminology component of modelling languages*) describing the relationship among modelling concepts available in enterprise modelling languages. They describe the concepts used, their properties and relationships, as well as some basic constraints, such as cardinality constraints.
- Ontological Theories defining the meaning (*semantics*) of enterprise modelling languages, to improve the analytic capability of engineering tools, and through them the usefulness of enterprise models. Typically, these *theories would be built inside the engineering tools*.

Ontological theories are formal models of the concepts that are used in CE model representations. They capture rules and constraints of the domain of interest, allowing useful inferences to be drawn, to analyze, execute (for example, simulate), cross check, and validate models.

5. Ontology

Research on ontology is becoming increasingly widespread in the computer science community, and its importance is being recognized in multiplicity of research fields and applications areas, including knowledge engineering, agent-based system design and enterprise integration (see overview in (Guarino, 1998). More recent definition of ontologies was proposed in (Gruber, 1995): "Ontologies are agreement about shared conceptualization. Shared conceptualization includes conceptual frameworks for modelling domain knowledge; content-specific protocols for communication among inter-operating agents; and agreement about the representation of particular domain theories. In the knowledge sharing context, ontologies are specified in the form of definitions of representational vocabulary. A very simple case would be a type hierarchy, specifying classes and their subsumption relationships. Relational database schemata also serve as ontologies by specifying the relations that can exist in some shared database and the integrity constraints that must hold for them".

Below we will use a following definition of ontology: "An ontology is a formal description of entities and their properties, relationships, constraints, behaviours".

Kinds of ontologies (Guarino, 1998), according to their level of dependence on a particular task or viewpoint. Top-level ontologies describe very general concepts like space, time, object, event, action, etc., which are independent of a particular problem or domain. Domain ontologies and task ontologies describe the vocabulary related to a generic domain or generic task or activity, by specializing the terms introduced in the top-level ontology. Application ontologies describe concepts depending both on a particular domain and task, which are often specialization of both related ontologies.

In (Bradshaw, 1992; Gruber, 1991; Gruninger, 1997) a set of integrated ontologies for representing enterprises spanning activities, states, time, organization, resources, and products is proposed. The approach to engineering ontologies begins with defining an ontology requirements; this is in the form of questions that an ontology must be able to answer, and is called the competency of the ontology. The second step in the design of an ontology is to define its terminology: its objects, attributes, and relations. The third step is to specify the definitions and constraints against the terminology, where possible.

6. Object-oriented dynamic constraint network as top-level ontology paradigm for CE configuring

Widely accepted that E&M activities can be regarded as search involving techniques to satisfy constraints (Giachetti *et al.*, 1997; Hirsch, 1995; Tsang, 1991). Constraint satisfaction is a fundamental problem among CE problems. Conventional constraint satisfaction procedures are designed for the problem with one constant set of constraints. For example, in engineering for manufacturing systems (design for productivity, configuration, layout, and scheduling), it is often necessary to solve a dynamic constraint satisfaction problem where the applicable constraints depend on design aspects (Smirnov, 1994). Ordering of constraints (priorities of aspects) is one of the most important mechanisms, which leads to the problem solution. There exist constraints hierarchies for real E&M problems.

The domain knowledge model of CE contains entities (objects), which can be of different types (classes). The multi-level and multi-aspect/viewpoint are used for PPR-model description. Obviously, each user (aspect) works with his own ontology-oriented constraints network. KM tools support the conversion of PPR-model from an ontology to another one.

An abstract PPR-model is based on the concept of ontology-oriented constraint networks. General ideas of networks are represented by the concept of multi-ontology classes of entities, the logic of attributes and the constraint satisfaction problem model. This abstract model unifies the main concepts of languages such as standard object-oriented languages with classes, and constraint programming languages. This integration supports the declarative representation, efficiency of dynamic constraint solving, as well as the powerful problem modelling capability, maintainability, reusability, and extensibility of the object-oriented technology.

Ontology-oriented constraints network model is denoted A = *(St, C)*, *St* — an ontology structure, *C* — a set of ontology constraints. To deal with the concept schema of configuring process defined in terms of constraints, dynamic constraints network (DCN) model is applied. A static constraint network (SCN) $A_i = (V_i, D_i, C_i)$, involves a set of variables $V_i = (v_{i1}, v_{i2}, \ldots, v_{iN_i})$, each taking value in its respective domain $D_i = D_{i1} \times D_{i2} \times \ldots \times D_{ij} \times \ldots \times D_{iN_i} = \mathop{\times}\limits_{j=1}^{N_i} D_{ij}$, and a set of constraints $C_i = \{c_{i1}, c_{i2}, \ldots, c_{ik}\}$. A DCN N is a sequence of SCNs, each resulting from a change in the preceding one imposed by "the outside world". For description of top-level ontology the following abstract model based on integration of DCN model and object-oriented model (Royer, 1992) could be used.

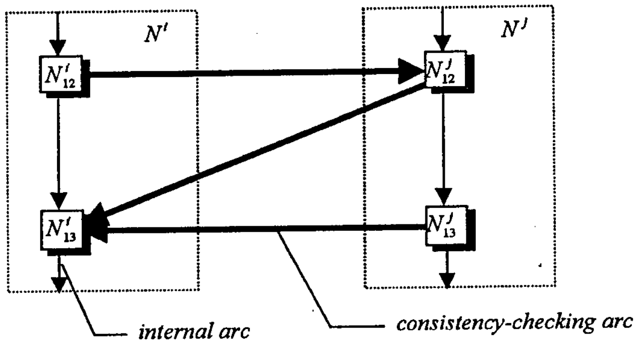
Model: $M_c = [<A_1, F_{11}, \ldots, F_{1n}>, \ldots, <A_n, F_{n1}, \ldots, F_{nn}>]$, A_i — an ontology, F_{ik} a conversion from an ontology A_i to A_k, F_{ii} — an identity. Ontology: $A_i = (V_i, D_i, C_i)$ is a SCN. Conversion: $F_{ik} = \mathop{\times}\limits_{j=1}^{N_i} v_{ij} = E_{D_{ij}} \rightarrow \mathop{\times}\limits_{l=1}^{N_i} v_{kl} = E_{D_{KL}}$, where $v_{ij} \in V_i$, $j = 1, \ldots, N_i$, $D_{ij} \subseteq D_i$, and $\mathop{\times}\limits_{j=1}^{N_i} v_{ij} = E_{D_{ij}}$ is a set $\{(v_{i1} = e_{i1}, \ldots, v_{iN_i} = e_{iN_i}) | e_{ij} \in E_{D_{ij}}\}$ describing the labeled Cartesian product, built on the structure (V_i, D_i). The term $E_{D_{ij}}$ designates the values set of D_{ij}. Constraints: $C_i : \mathop{\times}\limits_{j=1}^{N_i} v_{ij} = E_{D_{ij}} \rightarrow Boolean = \{true, false\}$. Instance: $<d_1, \ldots, d_n>$, where $d_i =< A_i, (v_{i1} = e_{i1}, \ldots, v_{iN_i} = e_{iN_i}) >$ is a description in ontology A_i, $< v_{i1} = e_{i1}, \ldots, v_{iN_i} = e_{iN_i} >$ — a tuple of equalities in $\mathop{\times}\limits_{j=1}^{N_i} v_{ij} = E_{D_{ij}}$.

The above Ontology Management approach is based on two mechanisms:

- Class inheritance mechanism is supported by inheritance of class ontologies (attributes inheritance) and by inheritance of constraints on class attribute values,
- Constraints inheritance mechanism for inter-ontology conversion is supported by constraints inheritance for general model (constraints strengthening for "top-down" or "begin-end" processes).

7. Integration of ontology and agent

The implementation of the basic principle for the CE, i.e. its collaborative nature, is based on procedure distribution between different users (or different agents). In this case it is more natural to represent the configuration KM as a set of



n^i_{km} - node (agent) associated externally defined predicate from DCN N^i_{km};

l^i_r - variable from $V^i \subseteq V$, which is consistent with shared DCN N;

i, j – ontology numbers, k - task number, m – a state number

Figure 3. Configuring process specification as a communication network

8. CE configuring problems as constraints satisfaction model

The constraints satisfaction model has been applied to a wide range of CE problems (see Figure 4). In order to realize the above concept schema appropriate heuristics are proposed: (i) An assembly operation is a connection between two parts/subassemblies together with a set of constraints. (ii) An assembly order given by access conditions is a precedence order between two or more assembly operations. An assembly process can be serial i.e. one part is assembled at a stage/a time, or partial parallel with subassemblies. (iii) Two parts are connected if the intersection of sets of their attributes is not empty. (iv) Operations not preceded by any operations and those not followed by any operations can be executed at the first and last assembly stages respectively. (v) Connected operations at assembly stage i are operations belonging to all stages preceding i and those of stage i.

The rules set for assembly planning was formulated as follows (Smirnov, 1998):

- Rule 1. A fuzzy relation between parts can be defined as: $R: P \times P \to [0,1]$, $R\{p_j, p_k\} = \mu(C_{jk}),\ \ p_j, p_k \in P$. In other words, parts p_j and p_k should be connected at stage i when $\mu(C_{jk}) \geq \alpha_i$. The constraints are satisfied when $\mu(C_{jk}) \geq \alpha_i$, where α_i is the system truth threshold (Giachetti *et al.*, 1997). An $\mu(C_{jk})$ value can be assigned to every existing assembly operation (the operation should be executed at stage i):

$$\mu(C_{jk}) = \left(1 - \frac{i}{s+1}\right)\frac{|C_{jk}|}{\max\limits_{j,k}|C_{jk}|},$$

i - the stage number, s - the total number of stages.

- Rule 2. If the operation $\{p_j, p_k\}$ is not a fixed operation between stages l and u, then its α value is in the range $[\alpha_u, \alpha_l]$. $R(p_j, p_k)$ is the degree of connectivity of parts p_j and p_k. At the α_i-cut of fuzzy constraints network (P, R) parts p_j and p_k are connected iff $R(p_j, p_k) \geq \alpha_i$.

A car assembly in case when $|C_{jk}| = const$ for $\forall j, k$ is shown in Figure 5.

Assembly operations are: *AD, BC, BD, DE* with associated sets of constraints for their attributes consistency. An order derived from assembly access conditions is: *BC* < *BD*, *BD* < *AD*. The assembly stages can be deduced from operations as follows: *BC*: stage1; *BD*: stage2; *AD*: stage3; *DE*: [stage1, stage3]. The corresponding μ values are: μ_{BC}:0.75; μ_{BD}:0.5; μ_{DE}:[0.25;0.75]; μ_{AD}:0.25. In result the following fuzzy constraints network is developed (Figure 6). The set of α-cuts of network is shown in Figure 7.

Then a formal description of the problem in term of resources dependency network could be provided. The two types of notes represent goods and CE participants (units). A network is a directed acyclic $graph(B, L)$, where

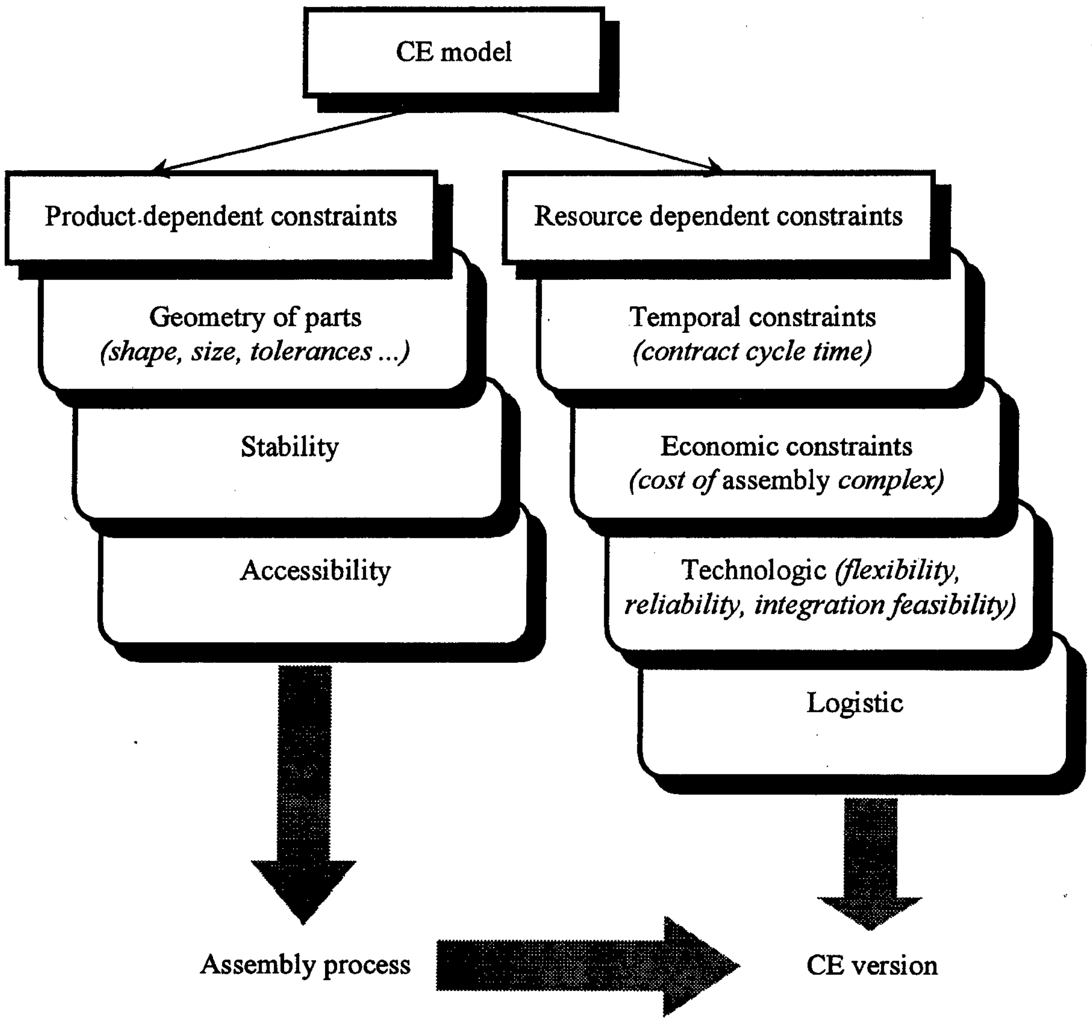
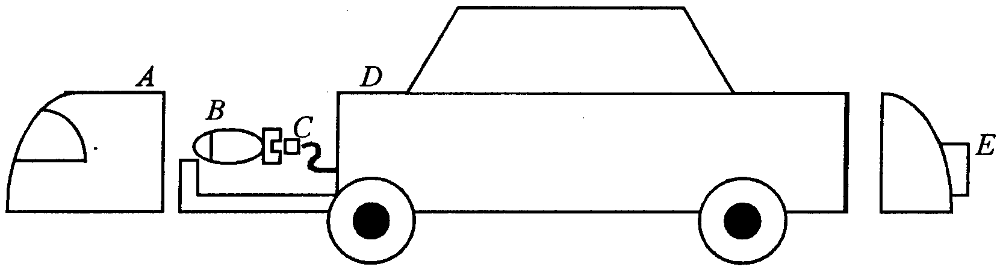

CE model
Product-dependent constraints
Resource dependent constraints
Geometry of parts
(shape, size, tolerances ...)
Stability
Accessibility
Temporal constraints
(contract cycle time)
Economic constraints
(cost of assembly complex)
Technologic (flexibility, reliability, integration feasibility)
Logistic
Assembly process
CE version

Figure 4. A constraints structure of CE model as "product-process-resource" mode



A - a cowling, *B* - a light, *C* -a cable, *D* - a cab, *E* - a trunk

Figure 5. A car assembly

$B = Un \bigcup G$, *Un* – a set of CE units (agents), *Un* - a set of producers, suppliers, customers, logistic companies, etc., *G* – a set of goods (parts, subassembly, products, orders, goals, etc.), *L* - a set of edges *L*, connecting units (agents) with goods they can use or can produce.

CE resources dependency network (see Figure 8) shows a result of conversion from CE Technology Model (see Figure 7) to CE Resource Model. In result a network that describes a connections of CE local tasks as the task allocation problem is developed. This problem could be solved by an agent-based method for decentralized task allocation (Walsh , 1998).

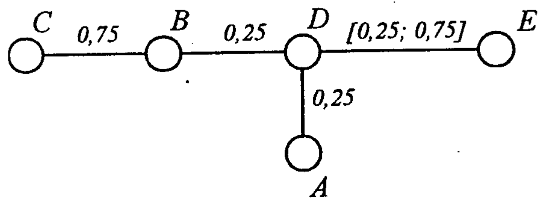


Figure 6. Fuzzy constraints network of the product «Car»

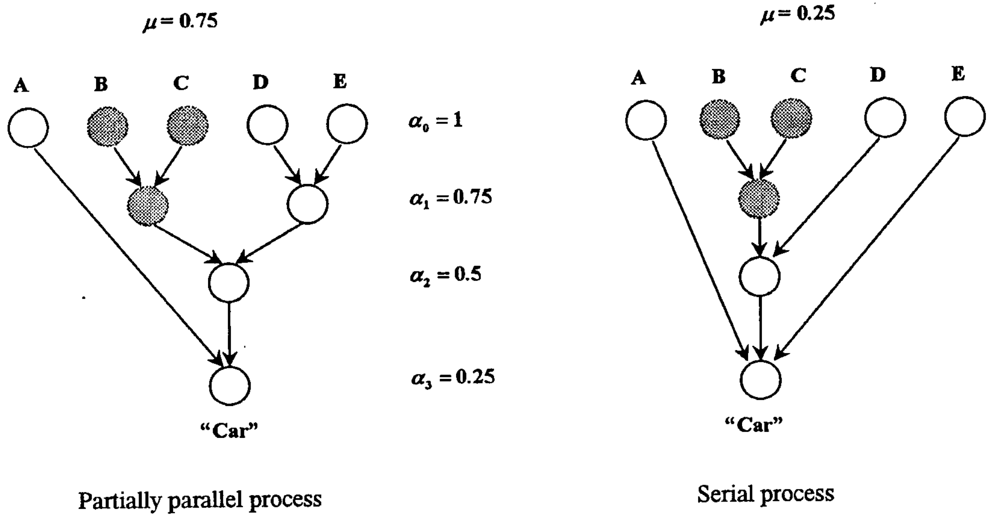
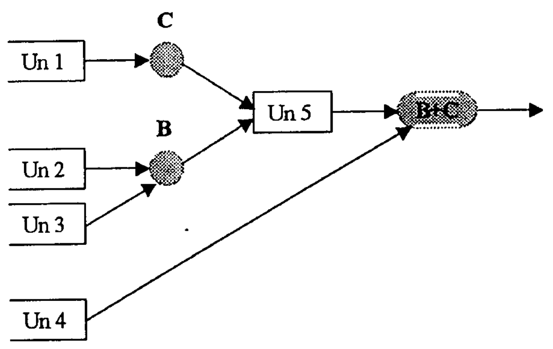


Figure 7. Assembly process organization



Un 1, Un2, Un 3 – produce units (Un 2, Un 3 – could produce same goods);
Un 4 – could trade unit for subassembly (B+C);
Un 5 – sub-assembly unit

Figure 8. An example of CE resources dependency network

9. DESO project: intelligent configurator

The goal of DESO (DEsign of Structured Objects) project is to develop a methodology and tools for automated re-use of industrial experience from large collection of knowledge&data in the engineering and business domains (Figure 9). DESO aims at establishing a knowledge platform enabling manufacturing enterprises to achieve reduced lead time and reduced cost based on customer requirements through customer satisfaction by means of improved availability, communication and quality of product information. DESO follows a decentralized method for intelligent knowledge and solutions access. Configuring process incorporates the following features: order-free selection, limits of resources, optimization (minimization or maximization), default values, freedom to make changes in CE Model. The CE model has an inheritance tree, in which each node corresponds to a component type (Figure 10). A component is described by a set of constraint variables that represents its attributes (such as its price or the resource it consumes or produces), its connections with other components, and its type.

A CE model is an objects library composed of the following:

- A set of object classes structured as a taxonomy, i.e., each object is linked to one or more other objects by a sub-class/super-class relationship.
- For each object class a set of relations is defined linking it to other object classes as well as a constraints set for each relation.
- For each object class a set of attributes plus a definition of the intended meaning of each attribute.

The "product-process-resource" model serves as a knowledge repository for manufacturing system (Figure 10).

A typical technologic PROCESS could be specified for PART class. Appearance of each operation in techologic process specifies a set of constraints, derived from part and operation attributes (see Table). When operations are

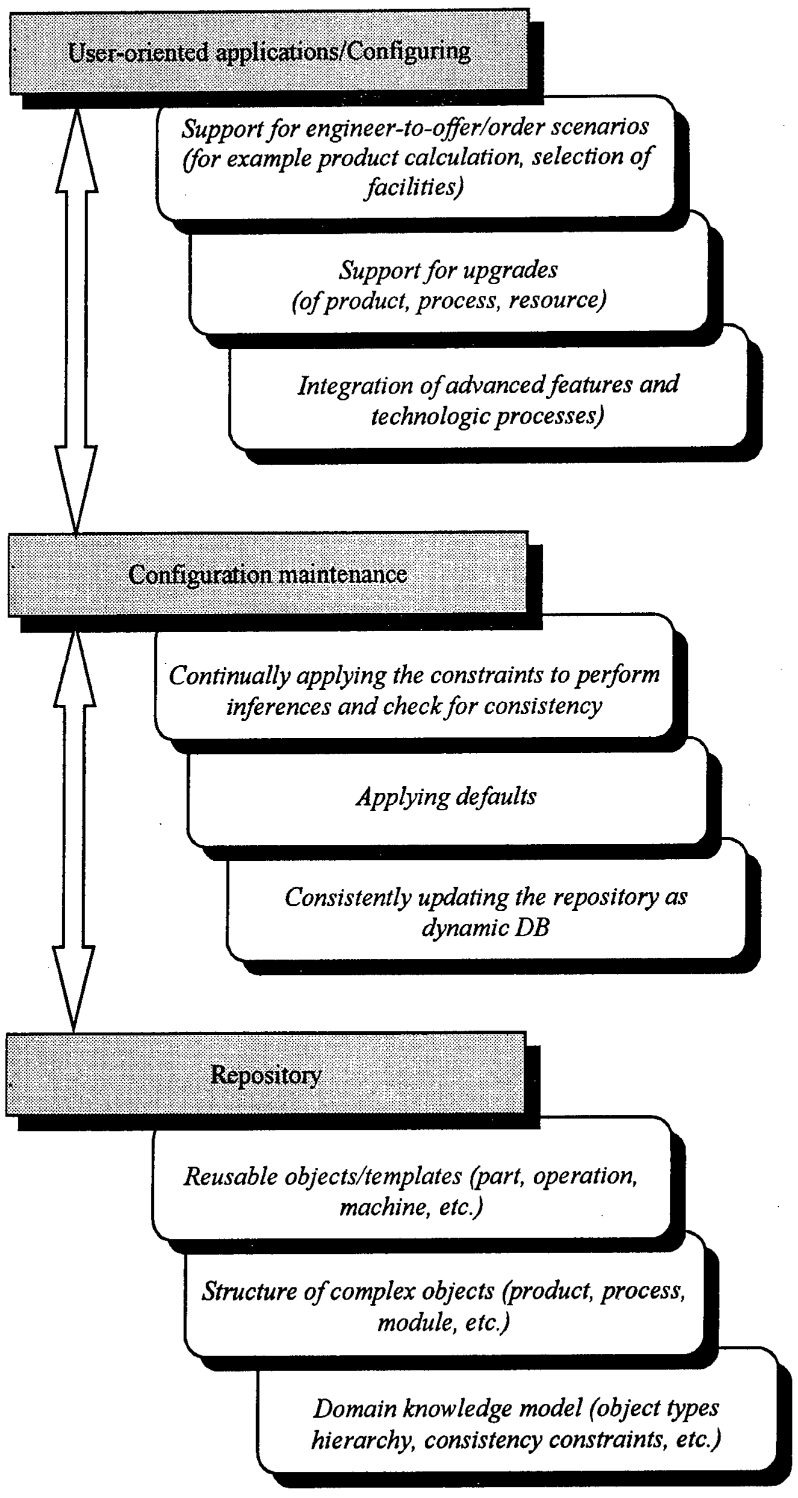

User-oriented applications/Configuring
Support for engineer-to-offer/order scenarios (for example product calculation, selection of facilities)
Support for upgrades (of product, process, resource)
Integration of advanced features and technologic processes)
Configuration maintenance
Continually applying the constraints to perform inferences and check for consistency
Applying defaults
Consistently updating the repository as dynamic DB
Repository
Reusable objects/templates (part, operation, machine, etc.)
Structure of complex objects (product, process, module, etc.)
Domain knowledge model (object types hierarchy, consistency constraints, etc.)

Figure 9. DESO architecture

specified machines could be selected to perform each operation. "Operation -> Machine" relation specifies a set of constraints and search criteria for machines search. Each relation transforms into the following expression: (A1 **OR** A2 **OR** A3 ...) **AND** (B1 **OR** B2 **OR** B3 ...) ..., where Ai, Bi – elementary constraints (ITEM), connecting operation and machine attributes. A weight system could be specified for ITEMs (importance of satisfying a given condition). In this case the search is performed with due account of constraints importance. At query execution users

Table: Relation between part and operation/A template of technologic process

Part	Operation	Constraints
Output_Shaft	Face & Center	Operation. Precision **better** Part.Precision **AND** Part.In-process material **in** Operation.Support material **AND** Part.L1 <= Operation.L
.	Cpl. Turning	Operation. Precision **better** Part.Precision **AND** Part.In-process material **in** Operation.Support material **AND** Part.L1 <= Operation.L1 **AND** Part.D1 <= Operation.D1
	Washing	Part. Weight *Count <= Operation. Weight
	Hardening	Part. Rigidity = Operation. Rigidity

Note: Count – a number of similar parts in one SUBASSEMBLY.

can activate or deactivate some constraints. For instance, current or prospective (previously planned for a following time period) set of machines. A time interval for search execution could be specified as well. As a result of query execution users receive a list of allowed machines. When the search does not arrive to any results, the system based on constraints generates requirements to be met by sought-for machines.

The descriptions in the model are composed from shared ontologies, i.e. sets of agreed-upon terms and formally described meanings of the CE configuring domain. It is worth mentioning that each user category works only with the objects of the corresponding classes of the model (Figure 11). This sharing also takes into account the fact that these tools do not necessarily share the same internal model as DNC. CE manager functions are associated with corresponding object classes: Goal specification: (product, process), Decomposition: (parts/subassemblies, technologies), Selection: (parts/ subassemblies, technologies, units), Assignment: (parts/subassemblies, technologies, units, CE life cycle stages).

A number of agents are associated with the DESO functional tasks (Figure 12). KM tools (relation manager, class manager, and type manager) are agents that assist a domain expert to define relations between the concepts of problem domain, and a knowledge engineer to define the subject domain structure. These agents are called 'assistants' or A-agents. CE configuring is based on multi-level decomposition and collaborative work in DESO. An agent is associated with each task of CE configuring, which is task-specific entity with limited knowledge about the global system status. We shall call this agent a 'designer' or D-agent.

At the current stage of the DESO project only Windows based systems were developed: D-Agents (Direct Costing Environment, implemented in Excel 97; PQE system for requirement specification, implemented in Visual FoxPro) and A-agents (KM tools, which is currently enhancing the main version written in Visual C++).

External requirements on the CE and the CE units from the market were obtained via Direct Costing system. In order to elaborate recommendations for determining changes in production program, the following tasks were carried out: analysis of what part of costs at different levels is fixed and what part is variable, as well as what products are profitable and what products are non-profitable. The results were transmitted to the requirements specification system (PQE), which determined the most significant CE components for re-configuring on the basis of integrated constraints on the general attributes (such goods as "investments", etc.). Determined component classes with associated constraints were used as inputs for the KM tools. Current configuration was supposed to be known to the system. Proposed solutions (changes in configuration) were once again evaluated with PQE system.

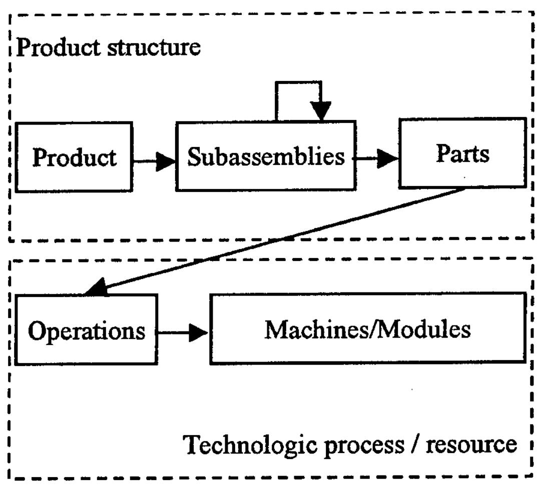
10. Conclusions

1. Constraint-based approach to the CE configuring suits well for different business processes from supply chain.

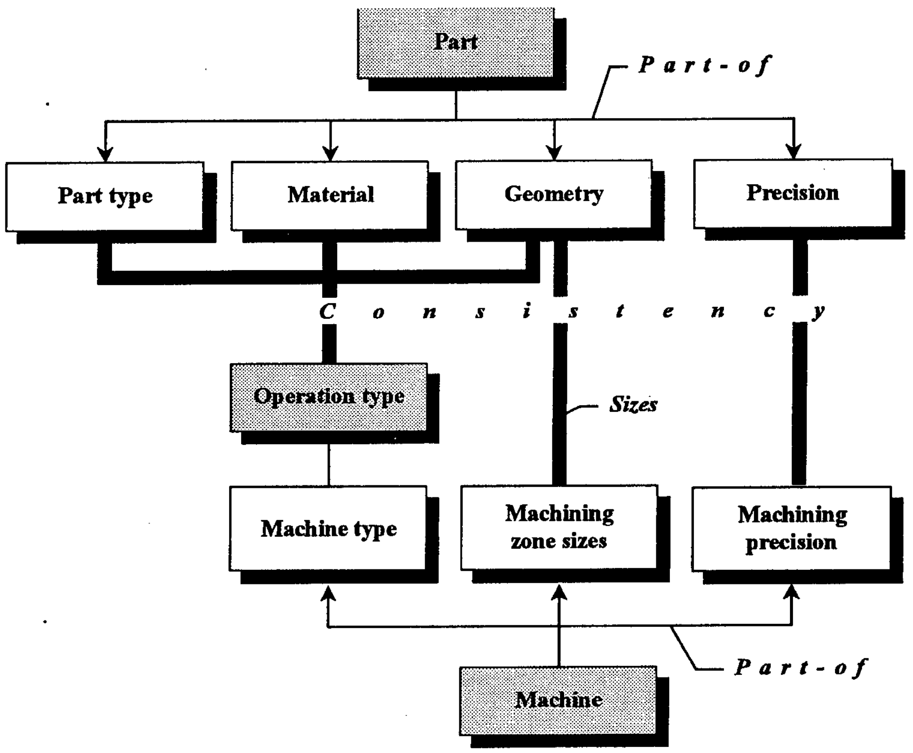
2. The universality of the knowledge representation by object-oriented dynamic constraints networks for all kinds of CE models makes it feasible to provide powerful interactive tool for the DB & KB maintenance

3. The knowledge management environment comprises means for CE project consistency control and allows multiple aspect/ontology-oriented collaborative engineering

a)



b)



a) a general knowledge domain model
b) an excerpt from the "Part – Operation – Machine" model

Figure 10. "Product-process-resource" model

4. Agent-based DESO architecture supports co-operation mode of interaction among agents, which reduces the time and increases the quality of CE configuring process

5. Constraint satisfaction mechanism gives us an opportunity to realise "test-generate" design methodology to prune problem search space that is more efficient as compared with the conventional "generate-test" approach

6. Agent-based knowledge management technology is an innovative technology in the domain of CE. Using this technology enables fewer people to make faster and better-quality decisions on CE configurations from CE unit templates under constraints networks with

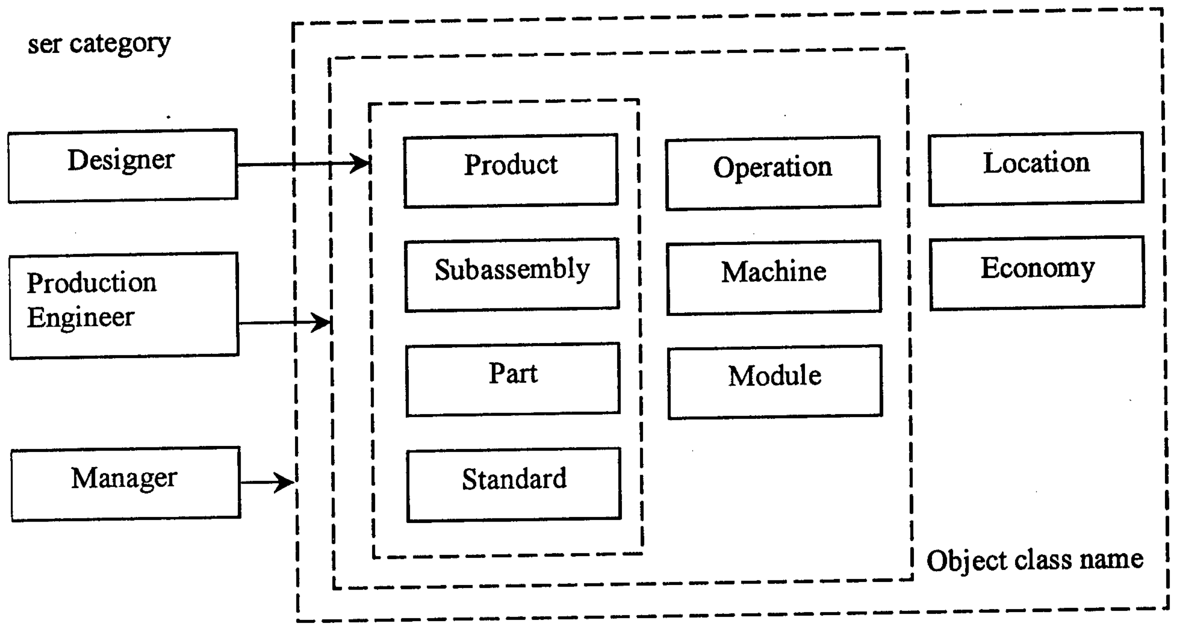


Figure 11. Sharing ontology example for the "product-process-resource" model

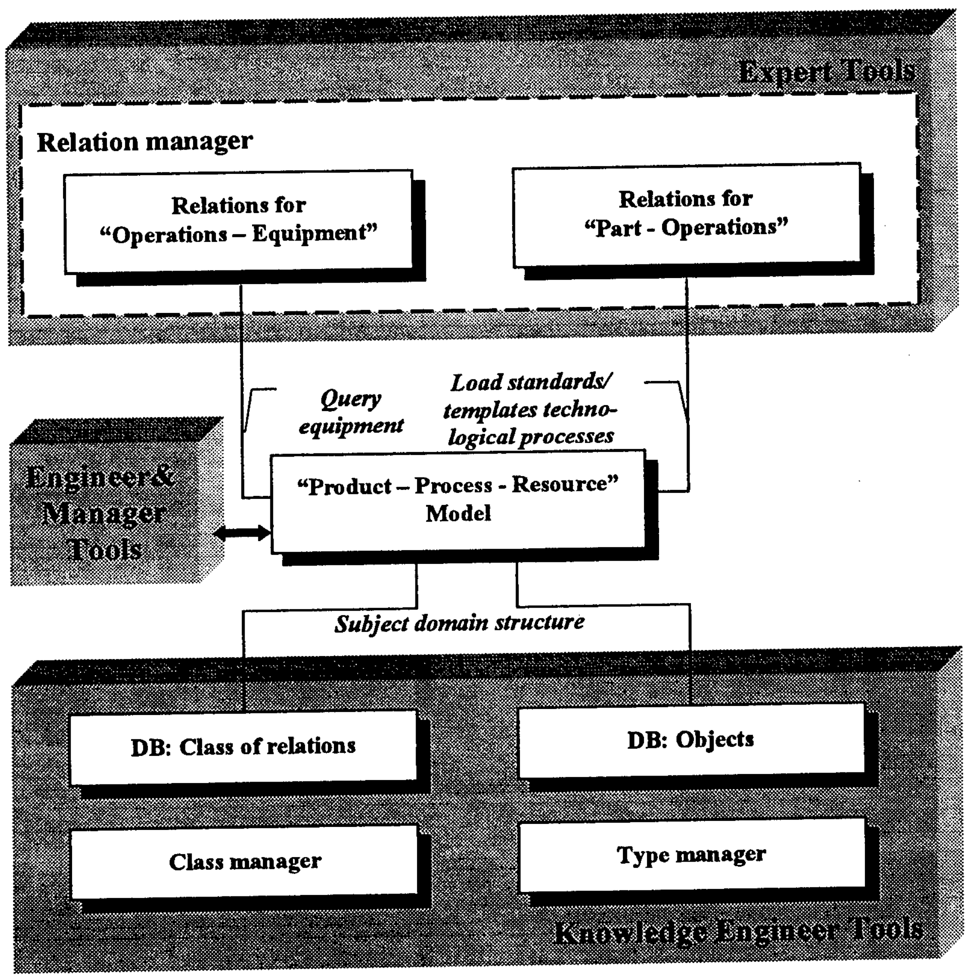


Figure 12. DESO functional architecture

reduced variance. Implementation of this technology will be followed by increased quality, reduced cost, reduced errors, decreased personnel required, better CE configuration solutions.

7. In the future we are going to development our approach under ILOG environment for CE reconfiguring.

11. Acknowledgements

This research was funded by the Ministry of Science and Technology of Russia (grants №№ 05.04.1233, 037.02.236.132/1-96, and 037.02.298.5/1-98) under the National Programmes: "Advanced Information Technologies" and "Computer Science in Russia" in 1996 - 1998. The DESO project would have never succeeded without the contribution of the programmers group of the laboratory of computer-aided integrated systems of SPIIRAS.

References

(Beyer, 1999) Beyer N., Frithjof W., Arrault J., Rodrigues F.. An Approach for a Practical Communication System for Supporting and Managing Concurrent Product Development. *Proceedings of the 5th International Conference on Concurrent Enterprising "The Concurrent Enterprise in Operation"* (Eds. by N.Wognum, K.-D.Thoben, K.S.Pawar), The Hague, The Netherlands, 1999, pp. 317—324.

(Bradshaw, 1992) Sharable ontologies as a basis for communication and collaboration in conceptual modeling / J.M.Bradshaw, P.D.Holm, J.H.Boose, D.Skuce, T.C.Lethbridge // Seventh Knowledge Acquisition for Knowledge-Based Systems Workshop: Proceedings. - Banff, Alberta, Canada, 1992. - Pp. 3.1 - 3.25.

(Bradshaw, 1993) Beyond the repertory grid: New approaches to constructivist knowledge acquisition tool development / J.M.Bradshaw, K.M.Ford, J.R.Adams-Webber, J.H.Boose / Eds. K.M.Ford & J.M.Bradshaw // Knowledge Acquisition as Modeling. - New York: John Wiley, 1993. - Pp. 287 - 333.

(Bussmann, 1998) Bussmann S. Agent-Oriented Programming of Manufacturing Control Tasks. *Proceedings of the International Conference on Multi Agent Systems*, Paris, France, 1998, pp. 57—63.

(CE-NET, 1998) CE NET White Paper on Concurrent Enterprising. Version 1.1. http://esoce.pl.ecp.fr/ce-net/CENET/XeditCENET.

(Delphigroup, 1998) www.delphigroup.com

(Donkin, 1998) Donkin R. Disciplines complete for a newcomer/ Financial Times, Wednesday, October 28, 1998, p. 14

(Fischer, 1996) Fischer K., Müller J.P., Heimig H., Scheer A.-W. Intelligent Agents in Virtual Enterprises. Proceedings of the First International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology, The Westminister Central Hall, London, UK, 1996, pp. 205—223.

(Giachetti, 1997) Giachetti, R.E., R.E.Young, A.Roggatz, W.Eversheim and G.Perrone. A methodology for the reduction of imprecision in the engineering process. *European Journal of Operational Research,* 1997, **100**, pp.277 – 292.

(Gruber, 1991) Gruber T.R. The role of common ontology in achieving sharable, reusable knowledge bases // Principles of Knowledge Representation and Reasoning / Eds. J.A.Allen, R.Fikes, E.Sandewall: Proceedings of the Second International Conference. - San Mateo, CA: Morgan Kaufmann, 1991. - Pp. 601 - 602.

(Gruber, 1995) Gruber T. Toward principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human and Computer Studies, 1995, 43 (5/6), pp. 907-928.

(Gruninger, 1997) Gruninger M. Integrated Ontologies for Enterprise Modelling. Enterprise Engineering and Integration: Building International Consensus k.Kosanke, J. Nell (eds.). Springer, 1997, pp. 368-377.

(Guarino, 1998) Guarino N. Formal ontology and Information Systems. Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998. http://www.ladseb.pd.cnr.it/

(Hirsch, 1995) Hirsch, B. Information System Concept for the Management of Distributed Production. *Computers in Industry*, 1995, **26**, 229 – 241. Elsevier Science B.V.

(ISO TC 184/SC 5/WG 1, 1997) "Requirements for enterprise reference architectures and methodologies" http://www.mel.nist.gov/sc5wg1/gera-std/ger-anxs.html]

(Jennings, 1996) Jennings, N., Faratin, P., Johnson, M., Brien, P.,n and Wiegand, M. Using Intelligent Agents to Manage Business Processes/ Proc. of Int. Conf. "The Practical Application of Intelligents and Multi-Agent Technology", London, 22-24 April 1996, pp.345-360.

(Livelink, 1998) Livelink: Collaborative Knowledge Management, http://www.opentext.com/livelink/ knowledge_management.html

(Nadoli, 1993). Nadoli, G. And Biegel, L. Intelligent manufacturing-Simulation Agent Tool (IMSAT). ACM Transactions on Modeling and Computer Simulation, vol.3, N 1, 1993, pp. 42-65.

(Neches, 1991) Enabling Technology for Knowledge Sharing / R.Neches, R.E.Fikes, T.Finin, T.Gruber, R.Patil, T.Senator, W.R.Swartout // AI Magazine, 1991. - V. 12, № 3. - Pp. 16 - 36.

(Patil, 1991) The DARPA Knowledge Sharing Effort: Progress Report / R.S.Patil, R.E.Fikes, P.F.Patel-Schneider, D.MacKay, T.Finin, T.Gruber, & R.Neches // The Annual International Conference on Knowledge Representation: Proceedings of KR'92. - Cambridge, MA, 1992.

(Pawlak, 1997) Pawlak, A., Cellary, W., Smirnov A., Warzee, X., Willis, J. Collaborative engineering based on the Web - how far to go?, in *Advances in Information Technologies: The Business Challenge* (eds. J.-Y.Roger, B.Stanford-Smith, P.K.Kidd), IOS Press, 1997, pp. 434 - 441.

(Royer, 1992) Royer, J. A new set interpretation of the inheritance relation and its checking. OOPS Messager. Vol. 3, N.3 , 1992, pp. 22-40.

(Sandholm, 1998) Sandholm T. Agents in Electronic Commerce: Component Technologies for Automated Negotiation and Coalition Formation. *Proceedings of the International Conference on Multi Agent Systems*, Paris, France, 1998, pp. 10—11.

(Scherer, 1999) Scherer R.J. and Katranuschkov P. Knowledge-Based Enhancements to Product Data Server Technology for Concurrent Engineering. *Proceedings of the 5th International Conference on Concurrent Enterprising "The Concurrent Enterprise in Operation"* (Eds. by N.Wognum, K.-D.Thoben, K.S.Pawar), The Hague, The Netherlands, 1999, pp. 121—128.

(Sheremetov and Smirnov, 1997) Sheremetov, L.B. and Smirnov, A.V. A Model of Distributed Constraint Satisfaction Problem and an Algorithm for

Configuration Design, in *Computación y Sistemas*, 1997, 2, 1, 91 - 100.

(Smirnov and Sheremetov, 1998). Smimov, A.V. and L.B.Sheremetov, Agent & Object-based Manufacturing Systems Re-engineering: a Case Study. In: *Changing the Ways We Work* (N.Mårtensson, R.Mackay and J.Björgvisson (Eds)). Proceedings of the Conference on Integration in Manufacturing, Göteborg, Sweden, IOS Press, 1998. Pp.369 – 378.

(Smirnov, 1994) Smirnov, A.V. Conceptual Design for Manufacture in Concurrent Engineering. In: *Proceedings of the Conference "Concurrent Engineering: Research and Applications"*, 1994. Pp.461—466. Pittsburgh, Pennsylvania.

(Smirnov, 1998) Smirnov A. Virtual enterprises: assembly planning based on fuzzy constrain satisfaction model. Advanced Summer Institute'98 ICINS – NOE: Proceedings of the International Conference ASI'98. Bremen, Germany, 1998. P. 37 – 38.

(Stumptner, 1997) Stumptner M. An overview of knowledge-based configuration. AI Communications, vol.10, n 2, 1997, pp. 111-125

(Tsang,1991). Tsang J.P. Constraint Propagation Issues in Automated Design. In: *Expert Systems in Engineering: Principles and Applications* (G.Gettlob and W.Nejdl (Eds.)), 1991, **462**, 135—151. Berlin, Springer- Verlag.

(Walsh, 1998) Walsh, W. and Wellman M. A market protocol for decentralized task allocation. Proc. of Int. Conf. on Multi Agent Systems, Paris, July 3-7, 1998, pp. 325—332.

COORDINATION IN MULTI AGENT SYSTEMS THROUGH FAIRNESS GUARANTEES

M. I. Smirnov

GMD FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
E-mail: smirnow@fokus.gmd.de Fax.: +49 30 34638000

Abstract.

Multi Agent Service Systems and main rules of their behaviour are introduced. MASS is formalised with two types of agents - service agents (static) and task agents (dynamic). Taxonomy of service agents is defined. Communication between task agents aims at co-ordination of tasks and service agents throughout a MASS, which is proposed to be best achieved via fairness guarantees. Fairness is two-fold: fairness with respect to all agents in MASS (agent fairness), and fairness with respect to all tasks in a MASS (task fairness). Both notions of fairness are formalised through a revenue, which is required to be fairly distributed among service agents based on trading of their revenue expectations for revenue offerings coming from tasks. The coordination component (relative coalition gain) of this trading process is carried by task agents. The MASS concept has been applied for the proposed group communication service Mediator, and in particular to its component which caches enriched IP multicast addresses and therefore facilitates highly dynamic large scale group communication.

Keywords: *coordination, fairness, caching, multi agent systems, multicast*

1. Introduction

Efficient and scalable solutions for coordination in Multi Agent Systems (MAS) will facilitate MAS based services in many areas where complex combinatorial problems should be solved, with data communication networks and group communication support in such networks being such an area and such a problem.

The following definitions are assumed throughout the paper. **Agent** - an entity designed to offer service[s] via its interfaces which is characterised by its behaviour [rules]. **Multi Agent System** (MAS) is a system in which agents co-operate to achieve a particular goal. **Multi Agent Service System** (MASS) is a MAS designed to offer a particular [set of] service[s] which are not available from any single agent otherwise.

While providing services the same agent could be a server or a client; we call them *service roles* of an agent. Further in a *server service role* of an agent we distinguish two options - *passive* server, traditionally awaiting for a service request from a client, and *push* server - pro-active offering of a service to a [set of] client[s]. In a MASS the desired application/end-user service is obtained as a result of agents co-operation and we assume that agents are not equivalent in their offered services. We further identify a richer taxonomy of agents in MASS.

Multi Agent Service Systems are enabling highly dynamic creation and dismantling of virtual *on-demand* enterprises [1] which is a typical *e-commerce* scenario. As first prototypes of such virtual enterprises already show there are two meta services which are critical for their mission: group communication service and federation service [2]. Group communication service for global virtual enterprises will use the Internet and its IP multicast technology [3]. However, as our analysis of non-IETF activities show [4] extensions are needed for 10 years old IP multicast service model [5], partly because large-scale multicast aware applications enabling virtual enterprises will be built on concepts of distributed processing environment. Therefore, an *addressed unit in a group* will have much finer granularity than existing currently MBone applications assume indirectly. Such an extension - Group Communication Mediator - is proposed in [4]. The Mediator is an active network node [6] to which group members (basically - objects) delegate group membership. The Mediator then will act on behalf of each member of a group. Note, that a group might be formed by the mediator based on a member profile. We consider a profile to be very similar to any policy description, i.e. constructed out of simple rules represented in pairs <condition, action>. Active networks allow also on demand service creation by means of mobile code. With respect to group communication services we also assume that some members of a group might wish to have specific code to be activated (on a node) for the time of their membership.

Let us summarise the issues leading to the concept of group communication service mediation (with IP multicast as underlying internetwork technology):

- Network Multicast Gain (resources savings) could be soon outperformed by savings in Information systems development (scalability obtained through grouping) and in decreased time-to-market for new services introduction;
- The issues of existing (low level) IP multicast which are delaying its wide deployment are lack of simple control over access, content, preferences, and address allocation and re-use;
- The issues listed above are all derivatives of a simple feature of IP multicast service model – group anonymity which prevents any kind of group ownership, and hence any kind of control;
- Different example – strong ownership over the group – is to be learned from a group communication paradigm adopted by ATM; however mapping of IP multicast to ATM point-to-multipoint connections implies that group management is to be outsourced to the multicast address resolution protocol (ARP) server;
- The outsourcing of ARP to a server co-located with a group management facility implies a strong opportunity to make rational use of multicast addresses enriched with user/application preferences (QoS, authentication, authorisation, accounting, platform specific and application specific parameters, etc.) while these addresses have been already acquired from the MALLOC[1] server;
- Making a step above opens an opportunity to achieve even better scalability and performance in group cOmmunication via co-ordinated behaviour of several active nodes (several mediators).

The complexity of a group communication service mediation in the Internet is a background model throughout the paper the rest of which is organised as follows. Section 2.1 tends to explain the framework and the vision adopted in this paper. Types of agents and a taxonomy of services performed by service agents are presented there. Section 2.2 further elaborates on this and introduces a service agent coalitions and rules of coalition establishment. Section 3.1 surveys main coordination principles adopted in this research, while section 3.2 explains how and why coordination in MASS is done through fairness and how fairness could be guaranteed. How service agents are made motivated to establish coalitions, and, therefore, to conform to the overall MASS strategy is explained in Section 3.3. Section 4 applies this framework and vision to the case of Mediator and further elaborates on its structure, components and evaluates the benefits of co-ordination in case of dynamic caching of multicast addresses enriched with object group profiles and also associated with run-time [mobile] code. The paper is concluded with a short summary and references.

1. MALLOC - Multicast address allocation working group of IETF engineering a standard Internet architecture

2. Agent Coalitions in MASS

2.1 Multi Agent Service System

The paper introduces MASS as synergetic multi agent systems performing a useful work (the notion of usefulness is however, outside of the paper's scope) better than with other approaches. We represent the usefulness by a flow of tasks entering MASS and specifically concentrate on how should agents be motivated to share the workload between themselves being self-organised in coalitions. Straightforwardly, we assume that the more tasks are performed in a unit of time the more value is gained by the MASS.

The main purpose of a Multi Agent Service System is to serve as a mediation facility for external systems (applications). That is, MASS is communicating tasks between consumer applications and, at the same time, performs a limited set of well defined services for these tasks. By this we see a MASS as a part of some workflow [7] external to MASS.

Based on the specification [8] we assume at least the following properties of agents rational behaviour:

belief means that, at least, the agent has a reasonable basis for stating the truth of a proposition, such as having the proposition stored in a data structure or expressed implicitly in the construction of the agent software;

intention means that the agent wishes some proposition, not currently believed to be true, to become true, and further that it will act in such a way that the truth of the proposition will be established;

uncertain means that the agent is not sure that a proposition is necessarily true, but it is more likely to be true than false. Believing a proposition and being uncertain of a proposition are mutually exclusive.

There are two types of agents within the MASS: service agents and task agents. MASS's *service agents* have statically assigned position within the physical topology underlying the MASS, while *task agents* are travelling across the service agents and carrying[2] task related information.

In this case service agents are representing what is known as *executive environment* for mobile agents. We use the term service agent, and not, e.g. agency while the notion of agency is tightly coupled with possible implementations (see e.g. [9]), another terminology option would be Stationary Agent and Mobile Agent, as defined by OMG [10]. However, by adjectives "service" and "task" we'd like to underline the main difference between the two types of agents in MASS (Fig. 1).

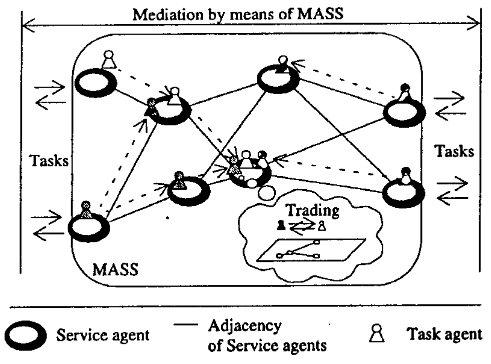


Fig. 1 The MASS Concept

Service agents are [quasi] statically positioned within the MASS topology and, in parallel to serving tasks, are running two types of protocols: neighbour discovery and capability discovery, so that at any moment in time each service agent with some degree of confidence is aware of what services are available at its neighbours. We formalise this in two general assumptions:

1. Assumption on *topological awareness* of agents: each agent in MAS is aware of its adjacent neighbours via periodical *neighbour discovery* message exchange. Neighbourhood relation is defined by a possibility to send/receive messages between adjacent agents without any other MAS entity being a mediator;
2. Assumption on *service awareness* of agents: each agent in MAS is aware of services provided by its adjacent neighbours via periodical *capability discovery* message exchange.

A taxonomy of services and service agents are dependent on external workflow specific features. While we are assuming computer communication networks to be the application area of MASS we define the following 4 major services:

- *source services* enabling an interface to a non-agent (real world) entity (application) wishing to utilise MASS mediation service, in particular a source service is responsible for creating needed number of task agents per incoming task;
- *provider services* controlling the overall constraints of the MASS and its common policies;
- *issue services* terminating the MASS operation for a task and. again, binding its results to a non-agent consumer entity (application), in particular an issue service is responsible for destroying needed number of task agents per performed task;
- *communication services* enabling needed instances of task agents to be in right service agents and served by right services.

The above services could be combined in every service agent and/or distributed between specialised service agents, which is assumed for the rest of the paper without any loss of generality, but for the sake of clarity.

Task agents are dynamic objects which are created by source service agents and destroyed by issue service agents. Task agents are communicating task information between service agents assuring their coordination for performing particular tasks. This communication is performed by physical transfer of a task agent from one service agent to another.

Task agents could be seen as complex messages (when on the fly) and as trading entities (when at service agent) at the same time. Task agents communicate to each other only when they are at the same service agent. Communication between task agents aims at coordination of tasks and service agents throughout a MASS.

We say that a number of service agents is forming a task coalition if they are involved in serving the same task. The coalitions are decided upon by service agents while they are actually facilitated by a flow of task agents.

2.2 Coalitions of Service Agents

We formalise the work applied to MASS by a flow of incoming tasks and specifically concentrate now on how should agents be motivated to share the workload between themselves via self-organising in coalitions. Straightforwardly, we assume that the more tasks are completed in a unit of time the more value is gained by the MASS.

Let us describe a task processing request in its part facilitating a coalition oriented collective behaviour of agents. Basically a request is a triplet $req = \langle T, \delta, \phi \rangle$, where T - is a task, δ - is a current value of relative coalition gain (see the definition below, in section 3), ϕ - a vector of Boolean variables with its i-th element set to true when the i-th service of task T is completed. We assume that T carries all the data needed to perform the task by a

2. This does not mean always *carrying* physically; in other cases it could be a simple tag with which a task agent is labelled and which tells a service agent where the task related information could be obtained from.

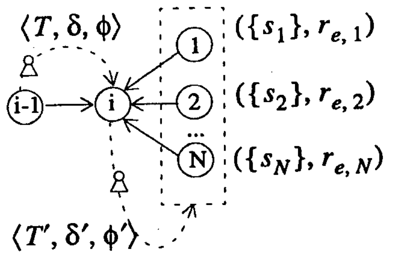
coalition of agents, including the value of revenue offered by a task - $T = T(\{S_T\}, R_\Sigma)$, where $\{S_T\}$ - is a set of required services for T, R_Σ - is a total revenue offered by a task to the MASS. As mentioned, R_Σ is a function decrementing its value over time.

How a service agent knows that participation in a coalition is good for it? This paper proposes a mechanism informing a service agent on what amount of revenue it will get if accepts and successfully performs a task. Note, that the maximal revenue for a service agent corresponds to the largest amount of resources available, i.e. to the cheapest[1] way to perform the requested service. If a task could be completed by a single agent then the revenue could be associated with the task and a service agent makes its decision based on a task description. If a single service agent can not complete the whole task, but is capable of making a part of it, it should also decide on the next agent willing to accept the uncompleted task. By making a reliable decision on a next service the current agent confirms its own part of the revenue, until the next service agent confirmation the revenue gained by the current one is conditional.

This kind of task sharing is particular true for the Internet: each IP datagram carries information needed for its processing in all IP modules along its path together with additional requests made by any of these IP modules to various servers comprising the Internet infrastructure. Such servers could provide Domain Name Service, Address Resolution Service, Classification, Address Translation, User Authentication, etc., and, finally, Accounting Service. In the example above these additional services share the time budget of a datagram, at the same time assuring the service completeness, and, sometimes, improving the quality.

Due to topology and service awareness assumptions for MASS we guarantee that coalitions are feasible and revenue considerations could be used for the optimization. Based on *topology and service awareness* a service agent creates its own understanding of a *service topology*. The notion of service topology does make sense only with regard to a particular task.

When receiving a triplet $req = \langle T, \delta, \phi \rangle$, an agent first, analyses the task T to see, whether it has needed services to perform a part of T, assuming the task arrived is uncompleted. If services are available, an agent sees, whether a task component value of revenue allows it to accept a task and decides, if applicable, whether it should participate in or establish a coalition with other agents to perform a task. This decision is based on the analysis of relative coalition gain value with regard to agent's current service topology understanding and policy constraints. For example, as it is demonstrated in Fig.2, right side, the current service agent (i) picks the cheapest next service agent offering a relevant service, i.e. the k-th one with maximal expected revenue ($r_{e,k}$), and forwards the task agent carrying the modified triplet $\langle T', \delta', \phi' \rangle$ to the k-th agent.



```
if ($\exists k$, $k$= 1..N: $\{s_k\} \backslash s_i \subseteq \{S_T\}$ & $\phi(s_k) = 0$)
then { $condR_i = (\delta + 1) \cdot r_{e,i}$ ;
       forward $\langle T', \delta', \phi' \rangle$
       to service agent k with $max(r_{e,k})$,
       $k = \overline{1, N}$;
       wait for *confirm* from k;}
if *confirm* then
       {$R_i = R_i + condR_i$; cache (s,k);}
```

Fig. 2 Coalition establishment example

The current agent waits for a confirmation from k that the service[s] has been successfully performed, and after that adds its own gain (up to this point being a conditional one - $condR_i$) to the total amount. The total revenue of current service agent is decremented by $condR_i$ and the k-th service agent is excluded from the service topology, when no *confirm* arrives from it. We assume that service agents are memorising successful coalitions in a service topology cache.

Cached coalitions should be aged by each participating service agent (the ageing interval estimation is a subject of task requirements). Alternatively, a coalition could be broken based on advertisement of the same service from a competing service agent and if a requesting agent believes this it could make a trial. We argue that a trial functionality is a powerful mean to make MASS scalable and efficient. Each server could send a probe task agent (assuming a repository of task agents) and verify the service provided by a potential partner agent.

A coalition is always dynamic and is represented by a tree or another structure, which always could be

1. This assumption is particularly true for many telecommunication services; e.g. the more bandwidth is provisioned on a link the smaller is the cost of its single unit.

decomposed into a number of trees. In general there are two concurrent flows along a coalition tree[1]: a bottom-up flow of service offers associated with revenue expectations from potential service agents and top-down flow of task agents which are carrying a cash revenue for best offers and motivate service agents to share this revenue. This motivation comes as a value of a relative coalition gain, multiplying the own revenue of each service agent by $(\delta + 1)$, where δ is introduced below.

3. Coordination Through Fairness

3.1 Coordination in MASS

Coordination in a Multi Agent Service System is a mean to achieve a desired system behaviour while each of the system components (agents) follows locally defined objectives which do not necessarily conform to the overall system strategy.
A strategy definition includes three components:

1. Distributed presentation of a current state;
2. Desired overall evolution of a system (to be reflected in agents intention), and
3. Primitive actions (services) leading in desired direction.

While it is well recognised that efficient coordination in MASS is a challenging task, it's even harder when applied to real-time systems, like computer communication networks. As, e.g. mentioned in [11], Real-Time multi agent systems are difficult to develop because additionally to response time constraints imposed on them they must support the evolution of their computational requirements. Evolution is a difficult requirement to meet, since not all the applications requirements are known in advance. However, following the commonalities of several coordination models, found in [12], we stick for MASS for the following few coordination principles:

1. Interaction via message exchange;
2. Scoping in message exchange, reflecting a service topology constraints.
3. Service advertising, service and capability discovery facilities (agent directory);
4. Rationality assumptions which serve as a basis for inter-agent communication:

We argue that coordination is to be considered as a separate part of MASS design, while it should be embedded into regular agents functionality during the MASS operation. This means merely, that agent's functionality design should be enriched with coordination concerns. An obvious way to do this is to make sure that local objectives are met by agents in a way conformant to the global MASS strategy. Local agent behaviour, however is not aware of global strategy. Therefore, there is a need to convey a global strategy to each agent. The best way to do this is to let each agent to know the relevant part of global strategy in a form which fits its local objectives representation. Coordination is thus proposed to be best achieved via fairness guarantees.

3.2 Fairness in MASS

Following the notion of fairness elaborated in [13] we consider the issue in settings where there are long living processes performed by agents which should be repeatedly scheduled for various tasks throughout the lifetime of a MAS. For any such instance a notion of desired load of a process could be developed, which is a function of the tasks it participates in. The unfairness of a system is the maximum, taken over all processes, of the difference between the desired load and the actual load.
The desired load of each agent is further formalised as a load which is maximising agents profit, taken in some cost units. This formalism is intuitively obvious: each action an agent is designed to perform is assigned a relative cost and expected revenue. If the action is offered to be performed by another agent then the cost sharing and revenue sharing take place. Fairness is two-fold: fairness with respect to all agents in MASS (agent fairness), and fairness with respect to all tasks in a MASS (task fairness). Agent fairness is defined as loading all agents with a fair share of tasks, task fairness is via providing a task with a fair service share from all agents. Both notions of fairness could be expressed through a revenue. Each task has a bounded, time dependent value of revenue[2] which potentially could be gained by a MASS. This revenue is to be fairly distributed among service agents based on trading of their revenue expectations for revenue offerings coming from tasks. The coordination component of this trading process is carried by task agents and is the current estimation of an optimal number of services for a given task. The optimal number of services will maximize the MASS revenue (and the revenue of a consumer system). The coordination component, called *relative coalition gain*, forces service agents to establish coalitions to serve particular tasks. This coalition enforcement is done via offering additional revenue which is maximal if a service agent follows a coalition rule offered by a task

1. Actually, both information exchanges take place, within adjacency, building, if possible a coalition tree step by step.

2. This potential revenue is assumed to be considered for co-ordination and mediation services only

agent. Coalition maintenance needs a specific protocol between service agents (section 2).

3.3 Coalition Motivation

To protect agents from a greedy behaviour we introduce a concept of *shared revenue*. Shared revenue is the basic motivation for agents to share the work load in a cooperative way. That is we assume that for a number of tasks N_t, there is an optimal number of agents performing these tasks $N_a^{opt}(N_t)$. The optimum condition is given by the following expression:

$$R_\Sigma(N_a, N_t) < R_\Sigma(N_a^{opt}(N_t), N_t)$$

for all $N_a \neq N_a^{opt}(N_t)$, where N_t - is a sample number[1] of tasks considered for optimization, N_a - is a number of agents participating in a coalition performing each task, N_a^{opt} - is an optimal number of agents in a coalition to perform a given number of tasks, R_Σ - is a total revenue gained by the multi agent system while performing a set of tasks.

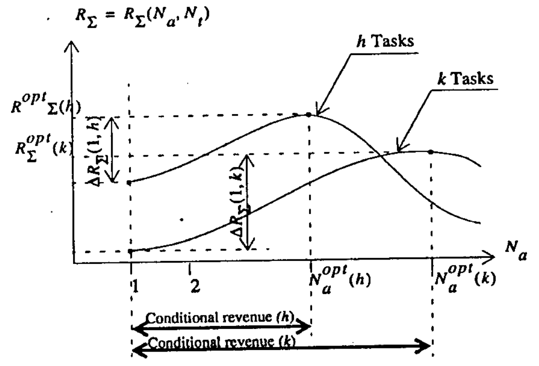


Fig. 3 Absolute Coalition Gain

The rationale behind this optimization comes from a consideration that efficient MASS should have agents specialised in performing particular actions, while end-user requests (tasks) will require several agents to be involved. Assuming partial overlapping of functionalities, and/or competition between agents we see that each task, requiring a timely performance, will be most optimally performed by a particular number of agents. The $R_\Sigma(N_a, N_t)$ is the generalisation of this fact for the case of different task rates (number of tasks per time unit) applied to MASS.

1. These tasks are considered to be concurrent, therefore this metric represents a *task rate* applied to a given MAS

Figure 3 shows hypothetical diagrams for a function $R_\Sigma(N_a)$ for two values of N_t - *h* and *k*. We define $\Delta R(N_t)$ - an *absolute coalition gain* - as

$$\Delta R(N_t, N_a) = R_\Sigma\left(N_a^{opt}(N_t), N_t\right) - R_\Sigma(N_a, N_t)$$

The coalition gain quantifies the increment of the revenue for the whole MASS when a [set of] N_t task[s] is performed by an optimal coalition of N_a agents in comparison with a smaller revenue when the same [set of] task[s] is performed by a single "greedy" service agent.

We further define a *relative coalition gain* $\delta R(N_t, N_a)$ as

$$\delta R(N_t, N_a) = \frac{\Delta R(N_t, N_a)}{R_\Sigma(N_t, N_a^{opt})}$$

The relative coalition gain (RCG) by definition is bounded:

$$0 < \delta R(N_t, N_a) < 1$$

The RCG (Fig. 4) is maximal for the first agent in the coalition chain and is exactly equal to 1 if this first agent gains nothing from performing the task (this hypothetical case would apply for a specific agent serving merely as an interface for incoming tasks - source service agent above). The RCG is minimal for the last service agent in the coalition and is always zero. This, however, does not mean that the last agent gains less than any other - absolute value of the personal revenue of an agent is independent of the RCG.

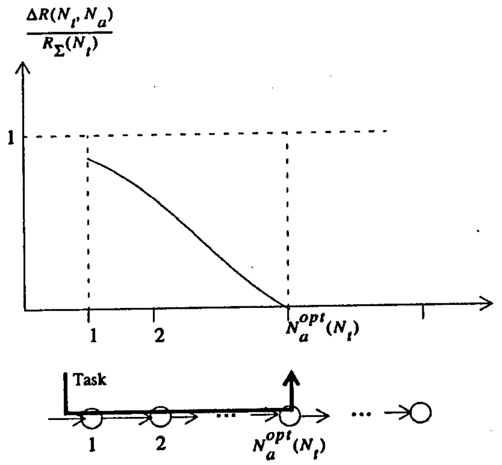


Fig. 4 Relative Coalition Gain

We can now relax the requirement to have an optimal number of agents to perform each task from the task flow of a given rate. The RCG definition is, however, still valid and useful if interpreted as a de-

sired probability of an agent to participate in a chain coalition while performing a task. Thus, RCG being a coalition participation probability, has intuitively very clear motivation: the more completed is the state of a task the less amount of choices could be applied by participating agents to form coalitions. In telecommunication networks, for example, if a task is a connection establishment then, obviously, a sending agent has much more choices to select a route to the destination, while the "last mile", has typically no alternatives.
We do not touch here the issue of hierarchical coalitions, however it seems obvious to extend the framework for the case of a hierarchy of coalitions. Another valuable interpretation of RCG comes for the group communication service. In this case the meaning of RCG is a motivation for e.g. shared multicast distribution trees, or co-ordinated use of highly restricted IP multicast address space.

4. MASS Application to Group Communication Mediation over IP Multicast

4.1 Mediator Overview

Up to this point we have introduced a Multi Agent Service Systems with a set of assumptions and motivations behind them. In Introduction we gave a summary of issues requiring an extension of IP Multicast service model adopted by the Internet. At the same time, more pragmatical considerations suggest that IP multicast by itself should be kept simple, however with complexity moved to policy control and authorisation, authentication and accounting entities. A valuable interpretation of RCG comes in this case as the motivation for shared multicast distribution trees. Shared revenue turns to be a valid mechanism to decide on many issues of IP multicast, like shared vs source specific trees, multicast group dynamics and subsequent tree rebuilding. A complete list of this issues could be found in [14], while the foundation of IP multicast as performed by what is called above "Task Agents", was first published in [15] and then further elaborated in [16]. We, however, concentrate here on a new aspect - dynamic co-operative caching of enriched IP multicast addresses, while it is a core of a proposed Mediator. We see group communication services mediation as one of active node facilities residing atop of fast IP forwarding engine controlled by a routing manager.
A Mediator maintains so called active cache with group membership and address information being currently in use and a passive cache. We further concentrate on a passive cache because our proposal is to realise this passive cache and subsequent dynamic co-operative hoarding of IP multicast addresses as a MASS. In particular, we treat further caches of enriched addresses as service agents; multicast flows - as task agents; service agents protocols - as service connectivity introduced below.

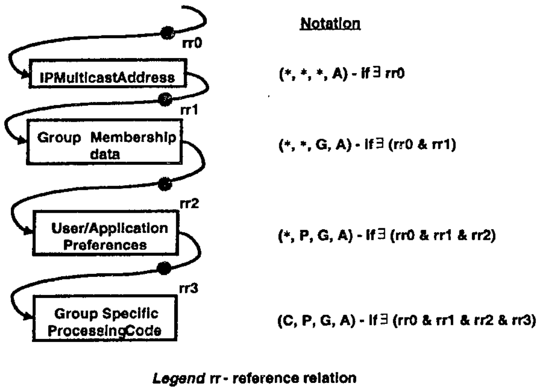

Notation
rr0
IPMulticastAddress
(*, *, *, A) - If ∃ rr0
rr1
Group Membership data
(*, *, G, A) - If ∃ (rr0 & rr1)
rr2
User/Application Preferences
(*, P, G, A) - If ∃ (rr0 & rr1 & rr2)
rr3
Group Specific ProcessingCode
(C, P, G, A) - If ∃ (rr0 & rr1 & rr2 & rr3)
Legend rr - reference relation

Fig. 5 Enriched IP Multicast addresses

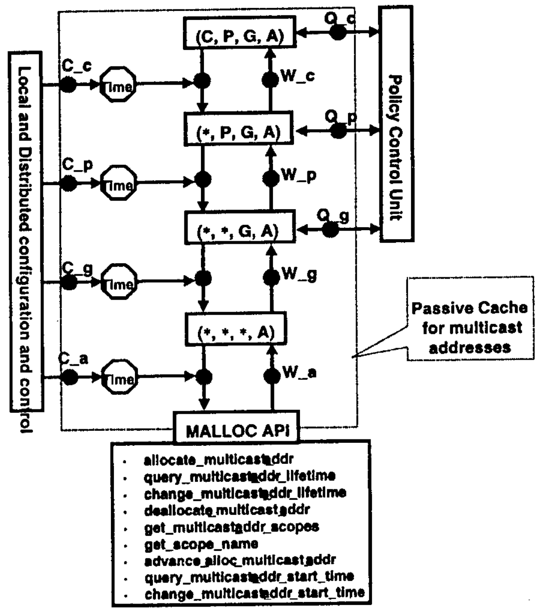
We propose to cache multicast group information in 4 main sets (Fig. 5) of enriched IP multicast addresses (presented in the ascending order of importance):

- IP Multicast addresses being currently in use or reserved for future use, denoted as group (*,*,*,A);
- IP Multicast addresses bound with groups of objects, denoted as group (*,*,G,A);
- IP Multicast addresses bound with groups of objects for which user/application preferences (profiles) have been specified, denoted as group (*,P,G,A);
- IP Multicast addresses bound with groups of objects for which user/application preferences (profiles) have been specified, and also some executable code has been loaded to support these preferences, denoted as group (C,P,G,A).

We further propose to define 4 reference relations (rr0 .. rr3) between these data sets:

- rr0 – shows that a particular multicast address has been acquired from MALLOC service and could be used by AN;
- rr1 – shows that a particular group of objects participating in group communication (users, applications, network interfaces, software objects, without loss of generality we hereafter will call any group an object group) is being associated with the multicast address (Note: this relation practically is the ownership relation);
- rr2 – shows that object group has some preferences associated with it (QoS parameters, AAA settings, hardware, platform- or application-specific preferences);
- rr3 – shows that in support of the above mentioned preferences there exist some executable code which has to be activated to facilitate fully specified group communication.

Our main interest is in passive cache (Fig.6).


(C, P, G, A)
(*, P, G, A)
(*, *, G, A)
(*, *, *, A)
Local and Distributed configuration and control
Policy Control Unit
C_c
C_p
C_g
C_a
W_c
W_p
W_g
W_a
Q_c
Q_p
Q_g
Time
Passive Cache for multicast addresses
MALLOC API
allocate_multicastaddr
query_multicastaddr_lifetime
change_multicastaddr_lifetime
deallocate_multicastaddr
get_multicastaddr_scopes
get_scope_name
advance_alloc_multicastaddr
query_multicastaddr_start_time
change_multicastaddr_start_time

Legend **C_* - cooling reference point;**
W_* - warming reference point;
Q_* - query reference point

Fig. 6 Passive Cache

Each cached entry of every set is supplied with an ageing timer: when the timer expires the associated group entry is moved to the lower level of passive cache. Finally, when the timer expires for an entry in group (*,*,*,A), the entry is de-allocated using regular MALLOC API. Passive cache is used to provide with very low latency multicast address allocation enriched with object group binding, user preferences and executable code.

While we should not cache multicast addresses without proper use of them we introduce ageing time-outs being in total within the time budget allowed by the MALLOC API and general framework.

We refer to the process of ageing of enriched addresses as *cooling* (the fewer data items are associated with the multicast address the cooler it is). On the contrary, the process of associating of additional data items to the multicast addresses we call *warming*. Therefore, the top of passive cache in Fig.6 is considered to be hot while the bottom – cold (and ready for de-allocation).

There are 3 types of services provided by agents in a passive cache. A generic service type description is as follows (we assume *x, y, z* to be pointers to corresponding data):

{C | P | G} -type agent with a service to provide a specific {code | preferences | object group} associated with {<P,G,A> | <G,A> | <A>}, with interfaces:

- cooling (with parameter timer defined),
- warming,
- query interface to policy control unit,
- {execution | profile | bind} interface via which the {code pointer | profile | object group} is given to active cache and to routing manager,

the behaviour is described below by:

if cooling-timer(x) is over then cool(x);
if query(y) and y=nil then warm(y);
if query(y) and y <> nil then execute(y).

This information on a proposed scenario is sufficient to model the behaviour and analyse the benefits.

4.2 Analysis of Dynamic Co-operative Caching of Multicast Addresses

Figure 7 presents the main illustration for a model.

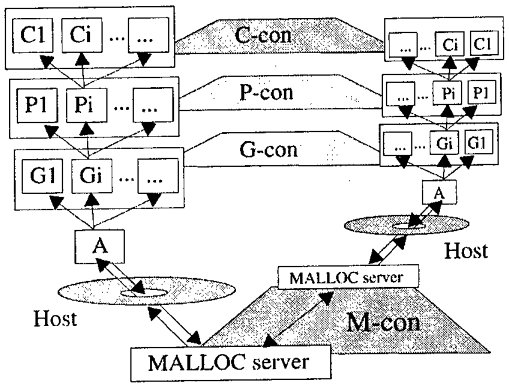


Fig. 7 Agents Hierarchy in passive caches

Each level of any passive cache *all over the network* is considered to be a server providing cached enriched addresses on request. This is a tree like hierarchy rooted at one or more MALLOC servers while all cached addresses are obtained from them. The tree specification template we consider is

hierarchy: {tree};
tree: {regular | irregular};
number_of_levels: L;
hierarchy_node_degree: g;
average_node_degree: d;
hoard_type: {uniform | non-uniform};
hoard_ratio: h;
hoard_sort: {individual | cooperative};
hoard_coop_horizon: {neighbours | all_in_scope};
hoard_coop_ratio: c.

<u>Scenario1: Individual caching.</u> Passive cache hierarchy for the whole network is a regular uniform tree: all caches at all levels behave the same way - they do individual caching (i.e. hoarding); for all levels *g = const; h = const; c = 0.* Assume that all used addresses *UA(k,L) = const = M,* then for all

(k,i) hoarded addresses $HA(k,i) = const = M \cdot h$, $i = 1..L$. Another important assumption here is that only *(C, P, G, A)* entries are requested from caches, that is only leaves of the hierarchy are really serving active caches.

All used addresses are given by $M \cdot g^{(L-1)}$ because there are $g^{(L-1)}$ leaves in the hierarchy. The number of allocated addresses by all the leaf caches is $M \cdot g^{(L-1)} \cdot (1+h)$.

Note that each server allocates $(1+h) \cdot AA_{(i-1)}$ addresses, where $AA_{(i-1)}$ are addresses allocated by its children. But the addresses allocated by the children are then "included" in the addresses allocated by the parent.

Therefore the utilization is: $Util = \dfrac{1}{(1+h)^{(L-1)}}$

Note, that the utilization for regular tree and uniform individual hoarding is independent of the degree of the hierarchy. Assuming *h*=0.333 the utilization is less than 0.5 for 3 levels in the hierarchy (Fig. 8).

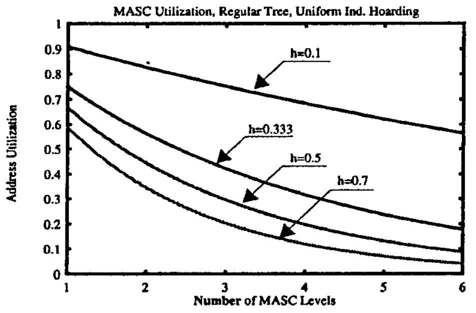


Fig. 8 Utilisation for scenario 1.

Scenario 2: Utilization boundary for regular tree and non-uniform individual hoarding. Caches hierarchy is the regular non-uniform tree: all caches at the same level of hierarchy behave the same way (they do individual hoarding BUT with values of "*h*" which differ from level to level). For all MAASs $g = const$; for all MAASs at *i*-th level $h(i) = const$; $c = 0$. Assume $UA(k,L) = const = M$, then for all *(k,i)* the following holds:

$$HA(k,i) = M \cdot h(i) \cdot g^{(i-1)} = const$$

We assume that $h(i) = h(i+1)^g$. The motivation for this assumption is as follows. Each leaf cache hoards *h*-th part of $CA_{(i+1,i)}$ - the addresses claimed from its parent, assuming that this amount will be sufficient to meet the pending demand from hosts. In case the demand from hosts at a leaf cache is more than $M \cdot (1+h)$ then this cache claims more from its parent (at level *L-1*) and so forth.

The analogy of *h(i)* with a probability for a *(k,i)*-th address allocation server to have claimed addresses available ($Pr\{Demand(i) > M \cdot h(i) \cdot g^{(i-1)}\}$) seems obvious. Assuming that these probabilities are independent for all children of an upper level *(i-1)* MAAS, we will have that this single parent could hoard much less: $h(i-1)^g$.

Applying the same considerations as in section 5.1 we obtain the following expression for the utilization:

$$Util = \frac{1}{\prod_{i \in [1,L]} \left\{1 + h^{g^{(i-1)}}\right\}}$$

Let us denote the expression above as ***T(h, L, g)***- it will be made clear below that all expressions for utilization with the non-uniform hoarding will have this expression as a multiplier.

Note that utilization for regular tree and non-uniform individual hoarding is dependent of the degree of the hierarchy (Fig. 9). This dependence is the largest for the values of *L* equal to 2 and 3. With the hierarchy degree *g=10*, *h(L) = 0.333*, and *L=3* the utilization is approaching 0.7, while even with *g=4* it's about 0.6.

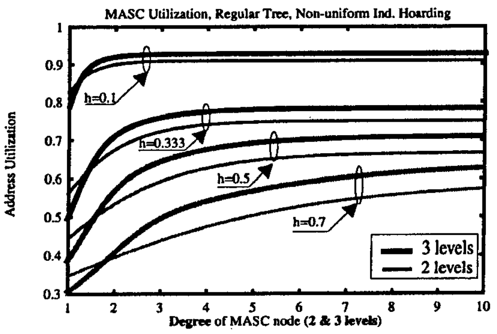


Fig. 9 Utilisation for scenario 2.

The *T(h, L, g)* expression is easy to interpret: utilization depends greatly on the cache level node degree, while the number of levels in the hierarchy influences the utilization marginally. The conclusion from this simple model could be as follows: for the better multicast addresses utilization it is much more important to provide a MAAS hierarchy with larger degree (3..5) than with more hierarchy levels.

Scenario 3: Co-operative caching. This model could be explained as follows. At level *L* each node of the cache hierarchy *really* uses *M* addresses and hoards $h \cdot M$ addresses, and additionally it hoards cooperatively $M \cdot (1+h) \cdot c$ addresses together with some other leaf caches. The *cooperation horizon* introduced above as the scope of cooperative hoarding is assumed to be "all children of the same parent".

Each cache at level L has $M \cdot (1+h)$ allocated addresses and all nodes together have additionally $M \cdot (1+h) \cdot c$ addresses. For the sake of utilization estimation we safely can assume that there is one more node in level L (a virtual, *(g+1)*-th one) which as if has all these cooperatively used addresses allocated. Then we can apply the model to the case when at the bottom level we have g nodes with the same number of allocated/hoarded addresses and *(g+1)*-th node which does not have really used addresses but only those which are "individually" hoarded (in fact - cooperatively hoarded).

For the *(g+1)*-th node $M = 0$; and $HA = M \cdot (1+h) \cdot c$.

The parent node at the level *L-1* sees that altogether all nodes below it have claimed $g \cdot M \cdot (1+h) \cdot (1-c) + c \cdot M \cdot (1+h)$ addresses. This parent node hoards the h^g -th part of the amount above. Note, that again, if there is a cooperative hoarding at the level *L-1* the total amount of individually and cooperatively hoarded addresses could be split into two parts - g nodes and the one virtual one which acts in the model as all nodes with regard to cooperative hoarding.

Therefore the following expression for the utilization applies:

$$Util = \frac{g}{g \cdot (1-c) + c} \cdot T(h, L, g)$$

where g - tree degree, c - constant cooperation ratio.

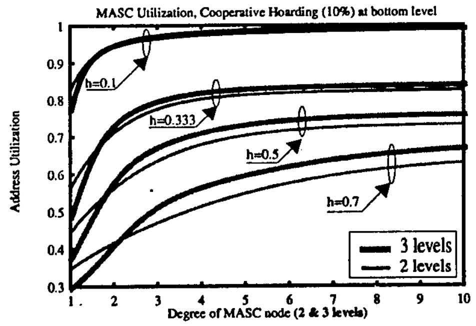


Fig. 10 Utilisation for scenario 3.

See a sample diagram in Fig. 10 which shows how the utilization is increasing with the degree of cache nodes (for 4 values of hoarding ratio and for 2 numbers of levels in the hierarchy) with the cooperative hoarding being introduced at only 10% of the total address space and only at the bottom of the caching tree.

5. Conclusions

Finally, let us highlight some benefits of the MASS approach as applied to group communication service mediation:

- Proposed scheme for multicast mediation fits well the Active Node paradigm and facilitates creation of next generation information systems, and dynamic virtual enterprises;
- Multicast mediation is efficient mean to meet the requirements of large scale deployment of IP Multicast even in presence of Firewalls, Proxies and network address translators,
- QoS issue is addressed exactly at the boundary of service level agreements;
- no interdomain multicast routing overhead in trusted domains;
- easily extensible set of policy rules controlling the mediation;
- very easy mapping to NBMA networks;
- multicast is feasible widespread while there is no need for address allocation/reallocation, etc.
- easily coexists with a classical RFC1112 model;
- address allocation being a critical service for large-scale IS could be efficiently realised via extending the MALLOC architecture with allocation cost and subsequent co-operative hoarding.

References

1. ADSS - Autonomous Decentralized Service System, cooperation project with Hitachi Ltd., GMD FOKUS, URL http://www.fokus.gmd.de/research/cc/platin/projects/adss/content.html
2. Autonomous Decentralized Service System (ADSS)Architecture, Hitachi Ltd., System Development Laboratory, URL: http://www.sdl.hitachi.co.jp/english/e-naiyo/e-seika1/e-s1.html
3. Deering, S., Host Extensions for IP Multicasting, IETF, RFC 1112, August, 1989
4. M. I. Smirnov, Cascaded Multicast Service in Active Networks,(unpublished memorandum), *submitted* to IWAN'99, 1st International Working Conference on Active Networks, Berlin, June 1999.
5. Deering, S., "Multicast Routing in a Datagram Internetwork", Ph.D. Thesis, Stanford University, December, 1991.
6. Tennenhouse, D.L. A Survey of Active Network Research, IEEE Communications Magazine, 35 (1), Jan., 1997, pp. 80-86
7. J. A. Gulla and O. I. Lindland, "Modeling cooperative work for workflow management," in Proc. of Advanced Information Systems Engineering (CAiSE), (Utrecht, The Netherlands), pp. 53-65, Springer Verlag, June 1994.
8. FIPA 97 Specification, Version 2.0 Part 2 Agent Communication Language, Foundation For Intelligent Physical Agents (FIPA), 23 rd October, 1998, URL: http://www.fipa.org/
9. Grasshoper. The Agent Platform. Technology Description, IKV++, 1997 - 1999, URL: http://www.ikv.de/products/grasshopper/index.html
10. MASIF-RTF Results, CORBA Facilities: Mobile Agent System Interoperability Facilities MASIF), Submission 4, the OMG document orbos/97-10-05, URL: http:///www.omg.org
11. J. C. Cruz, S. Tichelaar, O. Nierstrasz, A Coordination Component Framework for Open Systems Submitted to COORDINATION'97, Berlin, Germany, September 1-3, 1997, URL: http://iamwww.unibe.ch/~cruz/realtimecoord.html
12. W. C. Jamison, Approaches to Process and Agent Coordination , IBM Report, IBM Intelligent Agent Center of Competence, URL: http://www.cat.syr.edu/~wcjamiso/framework/framewor.html
13. M. Ajtai, J. Aspnes, M.Naor, Y. Rabani, L. J. Schulman, and O. Waarts. "Fairness in scheduling", In *Proc. of the 5th Ann. ACM-SIAM Symp. on Descrete Algorithms*, 1995, extended abstract available at URL: http://pine.cs.yale.edu:4201/home/soda95-abstract.html
14. M. Smirnov, A. Wolisz, IP Multicast Routing Through ATM Networks, Computer Communications, special issue "Internet: State-of-the-art", December, 1997
15. M. I. Smirnov, Object-Oriented Framework for the Multicast Applications Integration, In Proc. 1st Int. Workshop on Distr. Obj.-Oriented Computing, Frankfurt, Oct., 1995
16. M. I. Smirnov, Object-Oriented Framework for a Scalable Multicast Call Modelling, Lecture Notes in Computer Science, No. 1052, Springer, 1996

FORMAL PEPTIDE AS A BASIC AGENT OF IMMUNE NETWORKS: FROM NATURAL PROTOTYPE TO MATHEMATICAL THEORY AND APPLICATIONS

Alexander O. Tarakanov

St.Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences
E-mail: tar@mail.iias.spb.su

Abstract

This paper proposes an approach to multi-agent systems,based on mathematical modeling of behavior of proteins. Modern molecular biology shows, that such behavior play the key role both for immune and intellectual processes. Therefore, the paper presents mathematical notions of formal peptide and formal immune networks. Just like natural immune system, such multi-agent networks possess all the main capabilities of artificial intelligence. The paper considers also connection with the theory of linguistic valence, as well as real-world data mining applications to space navigation, ecology, medicine, etc.

Keywords: *formal peptide, agent, immune networks*

1. Introduction

Fluffy variety of modern information technologies, including multi-agent systems, sharply contrasts with the unify information basis of all living nature. This basis consists of universal genetic code and universal alphabet, which words are molecules of proteins.

Using multi-agent terminology, it can be said, that genetic code represents "messages" - instructions (programs), received by the cell from it's parent cell. In such sense proteins represent "agents" - biophysical mechanisms, which recognize and execute such programs. May be, that is the cause, why proteins are the most complex of the known molecules, as well as the most universal in their properties and functions.

Modern molecular biology shows, that proteins realize remarkably elegant, precise and reliable mechanisms of information processing, and such mechanisms play the key role both for immune and intellectual processes. For example, specialists call immune system (IS) "the second brain of vertebrates". Really, IS possesses the all main features of artificial intelligence (AI) system: memory, abilities to learn, to recognize and to make decision how to treat any macromolecule (*antigen*), even if the last one has never existed before on the Earth. In addition, the biological mechanisms of immunity are comparatively well studied, and in the same time the mechanisms of intellect remain a mystery. Especially interesting is the wide spread of the networks theory of immunity (N.Jerne), supposed that interactions between proteins of IS (*antibodies*) form the whole immune networks. The existence of such networks is beyond all doubts nowadays, because their fragments and interactions have been detected experimentally.

Based on the principles of IS, there arises a new and rapidly growing field of Artificial Immune Systems (AIS), offering powerful and robust information processing capabilities for solving complex problems. Like artificial neural networks (ANN), AIS can learn new information, recall previously learned information, and perform pattern recognition in a highly decentralized fashion. Treated as adaptive multi-agent systems, AIS have yet rather serious applications to security, faults detection, data mining, robotics, etc. [1].

However, comparing with ANN, the field of AIS hasn't yet its own mathematical basis. Nowadays AIS represent a set of heuristic algorithms, using ideas from genetic algorithms, cellular automata, etc. From the other side, role of proteins, as a basic natural agents of information processing, has been yet out of scope for multi-agent systems, including those of AI, ANN and AIS. Therefore, this paper tries to overcome such gaps. For this purpose we introduce a notion of *formal peptide* as a mathematical abstraction for key biophysical mechanisms of proteins' behavior. In the frame of interactions between formal peptides we determine networks of binding (recognizing), including *for-*

mal immune networks. We show, that even the simplest variants of such networks demonstrate important properties of immune response and immune memory. We consider also applications of the theory to analysis of global ballistic situations in near Earth space, complex evaluation of environmental and medical indicators, etc.

2. Mechanisms of behavior of proteins

In spite of exceptional complexity of phenomena, studied by molecular biology, they can represent a result of rather simple and general mechanisms, nevertheless, not so easy yielded to detection. A striking example represents discovery of double helix for chains of molecules, storing genetic code (1953). This spatial structure formed by week interactions between exclusively strictly adjusted molecular shapes, disposed in the same plane. No doubt, this is one of the most significant examples of geometrical correspondence on the biomolecular level.

But for proteins the search of mechanisms with analogous simplicity and explanatory power yet has no success, in spite of its long history, intensive efforts and even sensational statements. Nevertheless, there exist convincing evidence for the following principles:

1) function of any protein depends on its spatial conformation;

2) this conformation, in its own turn, is determined by the sequence of amino-acid's code of given protein. The first correspondence between spatial conformation and the functions of protein is realized by mechanisms of molecular recognition; the second correspondence between the code and the conformation of protein is realized by mechanisms of self-assembly. These mechanisms provide interactions between different molecules of proteins, as well as between different sectors of the same protein. Consider these mechanisms more detailed.

Molecular recognition. Under molecular recognition (further we'll denote it also as recognition) we'll understand interaction with high specific (selectivity) between the certain sectors of biomolecules. Such recognition forms the basis of several fundamental phenomena and it has admitted as one of the main problem of biochemistry.

As a result of molecular recognition emerges biological specific. The analogous phenomenon exists also in the non-organic nature - this is crystallization. Growing crystal is able to "extract" certain molecules from a heterogeneous solution and "reject" all the others.

According to "key and lock" hypothesis such phenomenon determined by a mutual adjusting of molecules, like fragments of mosaic. The basis of such adjusting form the weak interactions, which determine also the conformation of the most biomolecules. In addition, molecular recognition depends mainly on sizes and shapes of interacting surfaces, rather than on their chemical properties. However, last time an experimental data appear, which point out, that mechanisms of molecular recognition are much more complicated and they cannot be explained merely by shapes of the surfaces.

Self-assembly. Self-assembly (or *folding*) is understood as an ability of molecular chain of protein to assume the single conformation in space, in spite of an astronomical huge number of possible variants. Moreover, the time of folding is no more than the time of synthesis of protein's chain.

Such ability of proteins to fold in a unique way gives remarkably precise adjusting of structure to the function of protein. Proteins are built so precise, that even change of several atoms of single link in the chain can violate the structure and leads to disastrous consequences.

Mechanisms of proteins' folding are closely connected with the thorough problem of dawning order from chaos. Besides, the unique property is alternation of flexible and hard links along the protein's chain (so called "statistical balls" and "secondary structures"). Nowadays considered, that namely this property ensures folding of protein's molecule and makes this folding quick and faultless, as well as provides protein with the necessary mobility for its actions.

3. Formal peptide as a mathematical model

A notion of formal peptide (FP) has been introduced in [2] based on the above mechanisms of proteins' behavior (peptide is a small protein). This notion represents a mathematical abstraction for the biophysical principle of free energy dependence from the space conformation of protein's chain. Besides, FP unites the two main ideas:

- algebraic description of protein's chain geometry in 3D space by means of quaternions;
- definition of free energy as a function over elements of quaternions.

In the mathematical formulation FP is an ordered 5-tulp

$$P = < n, U, Q, V, v >,$$

which comprises the following components:

1) number of links $n>0$;
2) set of angles $U = \{ \varphi_k, \psi_k \}, k=1,\dots,n,$
where $-\pi \le \varphi_k \le \pi, -\pi \le \psi_k \le \pi$;
3) set of unit quaternions $Q = \{ Q_0, Q_k \}$, where quaternions $Q_k = Q_k(\varphi_k, \psi_k)$ defined by the formulas (2)-(3) and the resultant quaternion of FP Q_0 defined as their product:

$$Q_0 = Q_1 Q_2 \dots Q_n;$$

4) set of coefficients $V = \{v_{ij}\}, i=1,2,3,4, j\ge i$;
5) function v (without index), defined over the elements of the resultant quaternion Q_0 by the following quadratic form:

$$v = -\sum_{j\ge i} v_{ij} q_i q_j \,. \qquad (1)$$

In this definition unit quaternions $Q_1, \dots Q_n$ have the appearance

$$Q(\varphi, \psi) = q_1 H_0 + q_2 H_1 + q_3 H_2 + q_4 H_3\,, \qquad (2)$$

$$\left.\begin{aligned} q_1 &= -c_0 \sin\sigma \\ q_2 &= c_0 c_1 \cos\sigma + s_0 s_1 \cos\delta \\ q_3 &= c_0 s_1 \cos\sigma - s_0 c_1 \cos\delta \\ q_4 &= -s_0 \sin\delta \end{aligned}\right\}, \qquad (3)$$

$$c_0 = \cos\frac{1}{2}\theta,\ s_0 = \sin\frac{1}{2}\theta,\ c_1 = \cos\frac{1}{2}\eta,$$
$$s_1 = \sin\frac{1}{2}\eta,\quad \sigma = \frac{1}{2}(\psi+\varphi)\,,\quad \delta = \frac{1}{2}(\psi-\varphi),$$

where H_0 - unit matrix, H_1, H_2, H_3 - matrices of Pauli. Note, that the angles of FP θ, η correspond to the fixed angles of valence between atoms of nitrogen and carbon within the molecule of protein, and angles φ_k, ψ_k correspond to the angles of rotation of protein's chain around the axes of valence, called *torsion angles* in biophysics.
Consider values of torsion angles U as *states* of FP, and values of coefficients V - as *controls* of FP. Consider function v - as *free energy* of FP and introduce *configuration vector* of FP as such column-vector $[Q] = [q_1\ q_2\ q_3\ q_4]^T$, which elements are the elements of resultant quaternion Q_0. Consider also symmetric *energy matrix* $M(v)=[m_{ij}]$, $i,j=1,2,3,4$, determined by coefficients of quadratic form (1) as follows: $m_{ii}=v_{ii}$, $m_{ij}=m_{ji}=\frac{1}{2}v_{ij}$, $i\ne j$.
Hence, free energy of FP can be represented in vector-matrix form:

$$v = -[Q]^T M(v) [Q]. \qquad (4)$$

Define self-assembly of FP as a process of changing its states, according to the following system of differential equations:

$$\left.\begin{aligned} \ddot{\varphi}_k &= -\frac{\partial}{\partial\varphi_k} v - c\dot{\varphi}_k + f(h) \\ \ddot{\psi}_k &= -\frac{\partial}{\partial\psi_k} v - c\dot{\psi}_k + f(h) \end{aligned}\right\}, \qquad (5)$$

where k=1,...,n. These equations describe dynamics of changing states for bodies with unit masses under the influence of potential v, resistance of environment and disturbances. Here c - coefficient of sticky friction, $f(h)$ - some function, which gives stochastic disturbances with intensity, depends on the parameter h. This parameter simulate such factors, as temperature or acidity of environment, as well as any external factors.
The main interest for multi-agent systems represents not a process of self-assembly itself, but its main result - stable states of FP. Such states correspond to storing and retrieving of information. Evidently, under absence of external disturbances and rather sticky environment, stable states of FP, as well as transitions between them, are determined by the stationary values of the free energy, where its partitive derivation over any torsion angle is equal to zero. In addition, stable states are the stationary states corresponded to local minima of the free energy.
Therefore, stationary states of FP are determined by extremuma of quadratic form (1). The solving of this problem would be well known, if extremuma need to be determined only related to the elements of resultant quaternion. In such case the solution, connected with the representation (4), could be determined by the spectral factorization of energy matrix, formed by FP's controls. However, solutions of equations (5) ought to be determined relatively to the torsion angles, and the free energy of FP has nonlinear dependence on these angles. Moreover, equations (3) determine nonlinear bounds on the possible values of elements Q_0. Therefore, the special methods have been developed for the study of FPs' stationary states.
Therefore, FP described by a four real numbers (quaternion), which computed as a product of elementary quaternions, corresponded to the links of peptide (*amino-acids*). Every elementary quaternion, in its own turn, computed as a function over the two torsion angles of rotation. Besides, the FP's free energy computed as a sum of mutual

products of quaternion's elements with given weight coefficients.

As the result, we have torsion angles as inner parameters (states) of formal peptide, and the weight coefficient of the quadratic form - as outer parameters (controls). The values of inner parameters, which give local or global minimum to free energy, lower that some threshold, correspond to a stable state of formal peptide. The model demonstrates such important properties of proteins, as self-organized reaching of stable state (self-assembly), and its dependence from the number and the order (non-commutativity) of links. Besides, the transitions between the stable states of FP under an outer influence correspond to storing and retrieving information from memory.

4. Multiagent networks of bindings

The main condition for protein to function is its binding with another protein or another molecule. As a mathematical abstraction of this fact consider the natural spread of quadratic form (4) on the interactions between FPs [3].

Define *binding energy* between two FPs by the following bilinear form:

$$w(P,Q) = -[P]^T W[Q], \qquad (6)$$

where $[P]$, $[Q]$ - configuration vectors of the 1-st and the 2-d FP, correspondingly, W - *binding matrix*, $W=\{w_{ij}\}$, w_{ij} - given coefficients, $i,j=1,2,3,4$.

From this definition follows

$$w(Q,P) = -[Q]^T W^T [P],$$

and the direct representation of bilinear forms, determined by the matrices W and W^T, gives $w(P,Q) = w(Q,P)$ for any configurations. In other words, binding energy between two FPs is invariant to transposition of configuration vectors in the formula (6).

Define now *binding* as satisfying condition $w \leq w^\wedge$ for binding energy of FPs, where $w^\wedge$ - some threshold of binding. If this condition satisfied, then FPs considered as bound FPs and the following suppositions admitted:

1) bound FPs form the unite complex;

2) energy of complex for bound FPs is equal to the sum of their free energies plus binding energy:

$$v(P,Q) = v(P) + v(Q) + w(P,Q);$$

3) self-assembly (5) for every bound FP determined by the energy of their complex.

Hence, the stationary and stable states of the every bound FP also determined by the energy of their complex. Therefore, we've shown, that by binding and the subsequent breaking of bound FP can transfer to its other stable state. Such transforming of FP's state after binding we've called (formal) *allosteric effect*, because the real effects of the same name provide proteins-enzymes with abilities to perform their functions.

In addition, by allosteric effects some FPs get an ability to bind with those FPs, which they couldn't bind before. And new FPs can be involved in such process of subsequent binding. Based on this fact we've introduced the notion *network of binding*, which implies any subsequence of binding with allosteric effects. It has been proved constructively, that networks of binding exist.

For example, consider a special kind of FP, which depends from only one torsion angle φ. Consider a system of three such FPs: {P1, P2, P3}. Let they are in their stable states, correspondingly (see bold lines in Fig.1a):

$$\varphi_i^* = \frac{2\pi}{3}(i-1), \quad i=1,2,3.$$

Let by bind, any of this FP can change its stable state to the new one: $\varphi_i^{**} = \varphi_i^* + \pi$.

Define the threshold of binding $w^\wedge=0$ and the energy of interaction between Pi and Pj as:

$$w(\text{Pi},\text{Pj}) = -\cos\left(\varphi_i^* - \varphi_j^*\right).$$

Then no one pair of given FPs can bind.

Let the fourth FP, denoted by A (antigen), is added to the system. Let this FP is in its stable state $\varphi_4^* = \frac{\pi}{3}$ (dotted lines in Fig.1). This FP binds with P1: P1↔A (Fig.1a), and P1 changes its stable state (Fig.1b). Now P1 is able to bind with P2 or P3.

Consider, that P1 binds with P2 (Fig.1b) and P2 transfers to its new stable state (Fig.1c). Now P2 is able to bind with P3 (Fig.1c). If P3 change its stable state by this binding, then P3 becomes to its new stable state and is able to bind initial antigen A (Fig.1d).

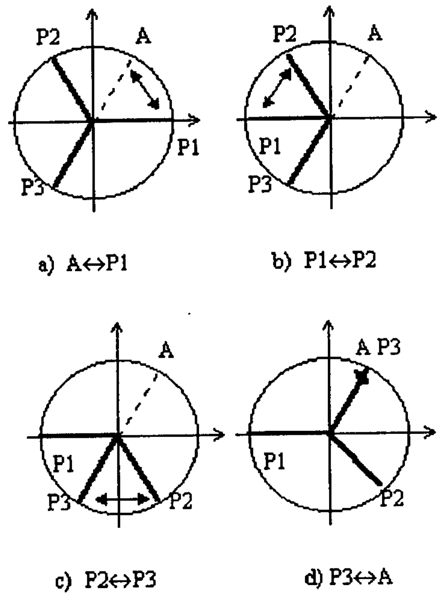


Fig.1. Diagram of bindings between FPs

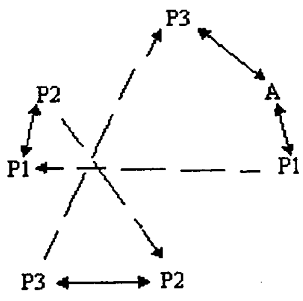


Fig2. Schematic representation of network of bindings

Described network of bindings between FPs: {A, P1, P2, P3} schematically showed in Fig2, where dotted lines denote transitions of stable states. Obviously, there exist also another ways of bindings between the given FPs. For example, initial activation of system {P1, P2, P3} by antigen A could be realized by binding A↔P2 (Fig.1a). Fig.1b,c also show, that there exist several variants of bindings. But the condition of the system, corresponded to Fig.1d, shows the only one way of binding.

5. Formal immune networks

For the modeling properties of immune networks we need to supply networks of binding of FPs with the possibilities, analogous to reproduction and death of cells. For this purpose we've introduce a notion of formal B-cell as a set

$$B = < P, Ir, Gv >,$$

which includes formal peptide P, indicator of regime Ir and map of controls Gv with the following restrictions:

1) B-cell can be in the regimes Ir={0,1,2}, where
Ir=0 - death: B-cell destroyed;
Ir=1 - regime of receptor: B-cell possesses the abilities of its FP - P;
Ir=2 - regime of reproduction: another one copy of B-cell created so, that for the both B-cells Ir=1 and their new control parameters determined by the map of controls Gv;
2) transition from the regime Ir=1 to the regime Ir=2 occurs only as a result of binding between FPs.

Define now formal immune network (FIN) as a network of bindings, which includes B-cells. Unlike cellular automata and artificial neural networks, which elements are fixed in space, the elements of FIN (B-cells and FPs) allowed to displace. We've studied the simplest variant of such displacement - linear displacement.

For this purpose a one-dimensional whole-number FIN defined with the following elements and rules.

Elements:

1) there are n sorts of FPs *{0,1,...,n-1}*, which denoted P0,P1,...,Pn-1;
accordingly, Bi denotes B-cell, which includes FP of the sort Pi;
2) a whole-number threshold of binding *n^* is given ;
3) energy of interaction between FPs defined by the formula

$$w(Pi,Pj) = \min \{ (i-j)\bmod(n),\ (j-i)\bmod(n) \} .$$

Rules of dispositions:

1) B-cells form one-dimensional sequence (population) without gaps, with begin (left) and end (right);
2) if cell Bj reproduces, then one of its copy remains on the former place, and the other copy added to the end of the population;
3) if cell Bj dies, then the other cells shift to the left and fill the gap.

We've introduced and studied the two kinds of one-dimensional whole-number FIN, called AB-networks and BB-networks.

AB-network AB(*n,n^*) defined as such one-dimensional whole-number FIN, which possesses, apart from B-cells, also the free FPs (antigens) of the n sorts A0,A1,...,An-1 with the following rules of displacement and interactions:

1) sequence of antigens displaced over the population of B-cells so, that to the every B-cell corresponded no more than one antigen;

2) interaction allowed only for a B-cell and an antigen over it by the following rules:
- B-cell dies, if there is no antigen over it, or if the energy of interaction between B-cell and antigen is higher, than the threshold $w>n^\wedge$;
- if $w=0$, then B-cell reproduces by forming two precise copies of itself (without mutations);
- if $0<w\leq n^\wedge$, then B-cell reproduced by forming two copies of its nearest sorts (with mutations);
- the result of interaction has no influence on the antigen;
- interactions realized consequently from left to right;
- when the end of population achieved, there performed return to its begin.

For AB-networks we've studied a homogeneous immunization, which complies the only one sort of antigen in it. We've proved constructively, that if even one B-cell bound antigen, then, after the finite number of steps, for the every antigen will be corresponded the B-cell of the same sort. This result affirms, that even the simplest variant of FIN shows the mechanisms, by which FPs (antigens) control the reproduction and the death of B-cells. Besides, we've determined the conditions of arising and supporting of formal immune response, which implies the B-cells' desire for acceptation of antigen's sort.

We've studied also a variant, where several sorts of B-cell generated and stored by interactions between B-cells themselves, in the absence of any antigen. For this purpose we've defined a notion of BB-network BB($n.n^\wedge$), as one-dimensional whole-number FIN with population of B-cells, formed by the following rules:
1) interactions allowed only for the neighbored B-cells with the numbers $2k-1$, $2k$, where $k=1,2...$ - a number of the pair of B-cells:
- if the last B-cell in population is odd (without pair) then it dies;
- if energy of interaction w>n^, then the second B-cell in the pair dies and its place remains free (Fig.3):

```
number of cell: 1 ... 2k-1 2k            1 ... 2k-1 2k
cell:           Bi ... Bm Bj ... Bk  ->  Bi ... Bm ♣ ... Bk
```

Fig.3. Death of B-cell Bj

- if energy of interaction 0<w≤n^ , then the second B-cell in the pair reproduces with mutations, where the first copy remains at the former place, and the second copy delayed (Fig.4):

```
number of cell: 1 ... 2k-1 2k            1 ... 2k-1 2k
cell            Bi ... Bm Bj ... Bk  ->  Bi ... Bm Bj-1 ... Bk
delayed copies:                                    Bj+1
```

Fig.4. Reproduction of B-cell Bj

- after all the pairs of population have interacted one time, B-cells shifted to the left for filling gaps of the died cells;
- then the delayed copies added to the end of population in order of increasing their numbers.

Evidently, a system of B-cells with such rules of interactions is determined, i.e. the initial population of B-cells determines completely all the further populations.

We've studied the possible variants of BB-networks' behavior. We've proved, that only one of the three regimes is possible for any initial population of B-cells:
1) death of all B-cells;
2) unbounded reproducing of B-cells;
3) cyclic reproduction of the initial population (formal immune memory).

In addition, we've proved the existence of such threshold $n^\wedge$ for any n, that BB-network BB($n,n^\wedge$) possesses a cyclic regime. We've pointed out also the variants of cyclic regimes with the several periods and dimensions of populations, including those, where the number of B-cells changes from population to population.

For example, in the network BB(4.1) any initial population of kind

BiBi+2Bi+3Bi

is cyclic with the period 2, and any population of kind

BiBi+2Bi+2Bi+3Bi+3BiBi+1

reproduced on the every step (with the period 1), where $i=0,1,...,n$. Specifically (for the convenience of description we'll also denote the sort of B-cell Bi as only the number i):

$$0230 \rightarrow 0331 \rightarrow 0230 \rightarrow \dots,$$
$$3123 \rightarrow 3220 \rightarrow 3123 \rightarrow \dots,$$
$$0223301 \rightarrow 0223301 \rightarrow \dots,$$
$$2001123 \rightarrow 2001123 \rightarrow \dots.$$

In the more complicated network BB($10,2$) for any sort $i=0,...,9$ there exist populations of kinds

Bi+3Bi-1BiBiBiBi+2
Bi+4BiBiBiBi+1Bi+2Bi+3 ,

which are cyclic with the period *3*. For example:

6233355→ 6323446→ 6224345→ 6233355→...
4999012 → 4980002 → 4890911 → 499012→...

In the same network any population of kind

Bi+2Bi Bi-2Bi

is cyclic with the period *4*, for example:

1979→ 187800→ 1770991→ 17980→ 1979→...

The obtained results show that even the simplest variants of FIN demonstrate such important effects, as:

- immune response in AB-networks under the control of antigens;
- immune memory and generation of new immune repertoire in the absence of outer antigen by means of the cyclic regimes of BB-networks.

It's worth to denote the main difference between FIN and cellular automata (CA). Namely, changing states of B-cell in FIN depends on possibility of its binding (recognition) with the concrete cell or antigen, but not on some total characteristic of cellular environment;

The principal difference between FIN and wide spread ANN is also determined by functions of their basic elements. While artificial neuron considered as a summator with given threshold, the FP, according to its biological prototype, ensures self-organization (self-assembly) of its parameters, as well as free binding with any other FP, depend on their reciprocal states.

Therefore, the proposed model permits to determine in a natural way a free energy of interaction between FPs as a bilinear form over the elements of corresponding quaternions. As a result of interaction, a binding (recognizing) of FPs occurs, if energy of interaction is low than some threshold; otherwise FPs don't bind. In addition, as a result of binding, the bound FPs can change their stable states. Namely such properties of FP admit to modeling the networks of interactions between FPs, including FIN.

6. A basic agent of linguistic models

It is known, that proteins represents the chains (words in alphabet) of 20 amino-acids. Usually, it pays no attention, that this number is approximately equal to the number of letters in the "classical" Indo-European alphabets. But similar analogy induces an idea, that FP can be treated also as a basic agent of linguistic models.

Really, consider an alphabet and connect to its every letter the FP with the fixed coefficients of quadratic form. Then the every word over such alphabet will determine the concrete value of the free energy of corresponding FP. Suppose, that the value of energy, which is higher than some threshold, determines the non-stability and decay of the corresponding FP. So, we've got, in general, an instrument for construction "correct words",- or *morphology*. Furthermore, determining bilinear forms of interactions between FPs, we've got an instrument for construction "correct sentences", - or *grammar*.

Of particular interest is the fact, that there exists rather advanced linguistic model of language, as if it has been especially developed for the proposed approach. The matter is of the *theory of linguistic valence*, developed by L.Teniere [4]. This theory stays somewhat isolated in linguistics, because it differ strongly from the wide spread generative grammars of N.Chomsky. Meanwhile, on the biological level it hasn't been detected nothing like "innate grammars", postulated by N.Chomsky. Moreover, the existence of such grammars itself imagined as rather problematical. At the same time, the theory of linguistic valence considers ability of a word to enter into syntactic connections with other elements of language based on the straight analogy of chemical interactions, even fixed in the name of the theory.

However, from the modern viewpoint it worth to make the definition of linguistic valence more precise. In fact, the chemical valence corresponds to the so called *strong links*. Their energy is much more higher, then the energy of the so called *week*, or *non-valence* links between molecules of proteins. Namely the week links, whose energy is comparative to the threshold energy of the thermal noise, correspond to the informational interactions. And namely such interactions determine in general the behavior of proteins. Hence, syntactic connections between words out to be corresponded, more precise, to the notion of non-valence interactions between FPs.

7. A basic agent of inference engine

Consider a set of FP sorts {S1,...,Sn}. Let two FPs can bind, if, and only if, they belong to the same sort.

Using such set of sorts, define a notion of formal T-cell <T> as a set of rules

$$Si_0 \rightarrow \langle T \rangle\, Si_1 \ldots Si_r\,, \qquad (7)$$

where $i_0, i_1, \ldots, i_r$ are indexes of sorts and belong to the set *1,...,n*.
The every rule (7) means:
1) T-cell has receptors of sorts $Si_1, \ldots, Si_r$ (right side of the rule);
2) T-cell synthesize FP of the sort Si_0 (left side of the rule), when all the receptors $Si_1,\ldots,Si_r$ are bound (T-cell *activated*).
Define now T-FIN as a set:

T-FIN = < Sorts, FPs, T-cells>.

Consider a set of concrete FPs: $Pk_1,\ldots,Pk_m$, belonged to the sorts $Sk_1,\ldots,Sk_m$, correspondingly. Consider also an antigen Ag, belonged to some sort Sj. Denote these facts by the following rules:

$$Sk_1 \rightarrow Pk_1\,, \ldots, Sk_1 \rightarrow Pk_1\,, \qquad (8)$$
$$Ag \rightarrow Sj\,. \qquad (9)$$

It can be shown strictly, that such T-FIN, added with the rules (8), (9), is equivalent to a special kind of attributive context-free grammar [5], where antigen Ag corresponds to the axiom of the grammar, sorts S1,..,Sn - to the nonterminals, FPs $Pk_1,\ldots,Pk_m$ - to the terminals, T-cells <T> - to the synthesized attributes. It worth to note, that rather powerful inference methods have been developed for such kind of grammars [5].
For example, consider an arbitrary triangle *ABC* in Fig.5. Denote its angles *A, B, C* and sides *A_B, B_C, A_C*.

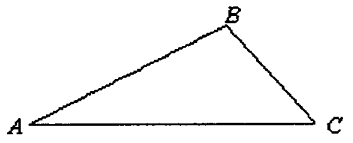


Fig.5. Triangle

Parameters of triangle are determined, according to the following theorems:
Angles theorem <Ta>:

$$A + B + C = \pi.$$

Sinus theorem <Tsin>:

$$\frac{A\_B}{\sin C} = \frac{A\_C}{\sin B} = \frac{B\_C}{\sin A}.$$

Cosinus theorem <Tcos>:

$$(B\_C)^2 = (A\_B)^2 + (A\_C)^2 - 2\,(A\_B)(A\_C)\cos A\,,$$
$$(A\_C)^2 = (A\_B)^2 + (B\_C)^2 - 2\,(A\_B)(B\_C)\cos B\,,$$
$$(A\_B)^2 = (A\_C)^2 + (B\_C)^2 - 2\,(A\_C)(B\_C)\cos C\,.$$

Hence, the following T-FIN could represent a model of arbitrary triangle:

FP sorts: A, B, C, A_B, B_C, A_C;
T-cells: <Ta>, <Tsin>, <Tcos>:
<Ta>:
A→<Ta>B-C, B→<Ta>A-C, C→<Ta>A-B,
<Tsin>:
A→<Tsin>B-B_C-A_C,
B→<Tsin>C-A_C-A_B,
C→<Tsin>A-A_B-B_C,
<Tcos>:
B_C→<Tcos>A-A_B-B_C,
A_C→<Tcos>B-A_B-B_C,
A_B→<Tcos>C-A_B-B_C.

Consider the following task:

find *B_C* by *C, A_C, A_B*. (10)

This task can be solved by T-FIN in the following way.
Firstly, given data activate the following rule of T-cell <Tsin>:

B→<Tsin>C-A_C-A_B .

Denote FP of the sort B, synthesized by this rule, as

B = <Tsin>C-A_C-A_B.

Secondly, this FP can bind receptor of the sort B. Hence, the following rule can be activated:

A → <Ta>B-C,

and T-cell <Ta> synthesizes FP of the sort A:

A = <Ta>B-C = <Ta><Tsin>C-A_C-A_B-C .

Thirdly, the last FP is able to bind receptor of the sort A. Hence, the following rule can be activated:

B_C→<Tcos>A-A_B-B_C

and T-cell <Tcos> synthesizes FP of the sort B_C:

B_C = <Tcos><Ta><Tsin>C-A_C-A_B-C-

A_B-B_C. (11)

It means, that T-FIN has synthesized the following solution of the task (10):
1) find angle B by given angle C and sides A_C, A_B by sinus theorem;
2) find angle A by known angles B and C by angles theorem;
3) find side B_C by known angle A and sides A_B, B_C by cosinus theorem.
Moreover, the solution of the stated task is given by T-FIN in the prefix Polish notation (11), which can be treated strictly as the program of computations [5].
Although the above geometry example seems rather simple, such kind of knowledge representation could be very useful, for example, for space navigation.

8. Data mining applications

8.1. Detecting dangerous situations in near-Earth space

Nowadays near-Earth space is being increasingly contaminated by hundreds of useless artificial objects which for all practical purposes could be called "space garbage". Space garbage presents a serious potential hazard to various space missions. Accurate hazard assessment of immediate space situations is crucial for mission safety. However, the assessment problem has not been solved on the global scale yet, primarily because of the following three difficulties:
1. Complicated rules of orbital motion, especially, relative motion of satellites
2. High order motion dynamics
3. Representation of 3D motion space on the 2D displays
Based on the FIN theory we've developed an applied approach to the analysis and visualization of global ballistic situation in near-Earth space. This approach uses also our results on the theory of orbital godograph, which reduces the dynamics of orbital motion of satellites to the relations between geometrical invariants in a specially constructed planes [6,7]. Within the frame of FIN theory such planes could be treated also as *shape spaces* [8] of FIN, where FPs model parameters of orbital godographs.
Applications of this approach together with the theory of orbital godograph facilitates successful resolution of the above mentioned difficulties within software thus providing the user with the essence of the assessment in comprehensive terms.
Appropriate FIN technologies could also be capable of providing real-time decision making in autonomous planning and control of space situation.
Consider, for example, the following task: over the set of all satellites and space garbage detect only those pairs, whose orbits intersects.
To solve this task represent every orbit as one-link FP (monopeptide) with quaternion unit vector

$$[Q]^T = [-\sin\omega,\ c_1\cos\omega+s_1,\ s_1\cos\omega-c_1,\ 0],$$

where ω- perigee argument (angle) of the orbit, $c_1=\cos\eta/2$, $s_1=\sin\eta/2$, η- FP's valence angle. Let every FP has an *active center* with an *active center matrix A*, and *environmental matrix S* is the same for the all FPs:

$$A=\begin{bmatrix} -e & 0 & 0 & 0 \\ 0 & c_1 e & s_1 e & 0 \\ 0 & s_1 p & -c_1 p & 0 \\ 0 & s_1\dfrac{1-e^2}{p} & -c_1\dfrac{1-e^2}{p} & 0 \end{bmatrix},$$

$$S=-\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

where p- focal parameter, e- excentricitet of the orbit.
Let binding energy between two FPs Q_1,Q_2 computed as

$$w(Q_1,Q_2)=-[Q_1]^T A_1^T SA_2[Q_2],$$

where A_1,A_2 - corresponding active centers matrices. Then, in accordance with equation

$$[Q]^T A^T = \begin{bmatrix} e\sin\omega & e\cos\omega & p & \dfrac{1-e^2}{p} \end{bmatrix},$$

we obtain the following result:

$$w(Q_1,Q_2)=2e_1e_2\cos(\omega_2-\omega_1)+ \\ +\frac{p_1}{p_2}(1-e_2^2)+\frac{p_2}{p_1}(1-e_1^2)$$

Let the threshold of binding is $w^\wedge=2$. Then, according to [6], bound FPs with $w\leq w^\wedge$ correspond to the crossing orbits, where $w=w^\wedge$ correspond to the touching orbits.

Giving the set of such FPs possibility to interact and to bind within FIN, we get, as a result of such self-organization, the complexes of FPs, which select from the all set of orbits only the crossing and the touching pairs of orbits. This approach has been spread also on the determination of spacecraft's visibility from the Earth point, as well as on the observation by spacecraft several areas on the Earth surface.

The computer's simulation over the set of more than 100 concrete orbits has showed, that such approach allows to represent clearly the global ballistic situation as a whole one, to analyze it quickly and to detect the dangerous objects as follows.

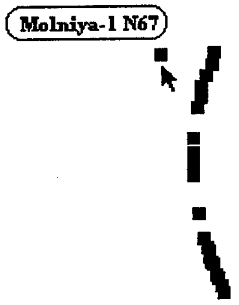
A fragment of standard data using for space navigation is represented in the Tab.1, where M- denotes satellite "Molniya" and parameters of orbits are: *p*- focal parameter, *e*- excentricitet, *ω* - perigee argument, *i* - angle to the plane of equator, *Ω*- ascending node longitude.

Tab.1

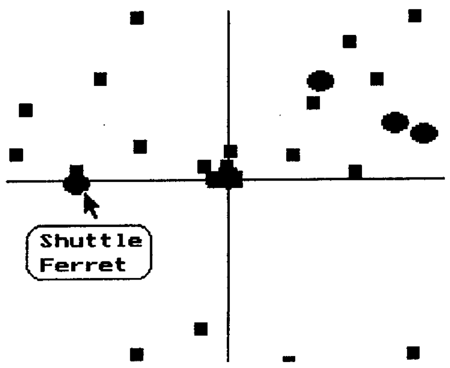
Parameters of orbits

Satellite, Time of perigee	p (km)	e	ω°	i°	Ω°
M-3 N43, 23 03 39	13079	0.71261	279	64.7	261
M-3 N52, 01 53 59	12670	0.72339	282	62.8	205
M-1 N67, 23 28 45	14396	0.68317	273	63.8	158
M-1 N71, 12 23 22	14289	0.67967	278	63.3	122
M-1 N78, 15 03 11	13467	0.70229	280	63.0	136
Shuttle, 22 56 44	6698	0.00042	285	49.6	123
Imuse, 11 10 38	42166	0.00008	79	1.3	131
Fleetsat, 15 40 56	42167	0.00027	168	0.6	42
Ferret, 02 06 52	6708	0.00755	43	97.4	331
...	...	...	...	...	...
...					

Computed by these data Fig.6a,b represent the fragments of global space situation mapped to the 2D shape spaces of FIN.



a)



b)

Fig.6. Shape spaces of FIN

In Fig.6a every orbit represented by FP (square) in 2D shape space of FIN. These FPs are able to bind according to there proximity in the shape space. Fig.6a clearly detects three groups of bounded FPs corresponded to the systems of communication satellites "Molniya-1,2,3". This FIN also clearly detects, that something wrong is with the satellite "Molniya-1" N67, because corresponded FP isn't able to bind with any other FP.

In Fig.6b every pair of intersecting orbits represented by the pair of bound FPs-square (according to the above presented formulas of active centers binding), and every pair of satellites positions on these orbits represented by another pair of FPs-circle. If FPs-square binds with FPs-circle and this complex binds with the horizontal line, then the two corresponding satellites have the dangerous approach. So FIN in Fig.6b detects among the all possible pairs of satellites the dangerous approach only between "Shuttle" and "Ferret". Such FIN is also well-suited for the prognosis of dangerous situations, because, unlike the complicated rules of orbital motion, FP-circles in Fig.6b are moving with time by the strict circles, and FP-squares don't change their positions at all.

This approach has been spread also on the determination of visibility of several satellites from the Earth point, as well as on the observation by satellite several areas on the Earth surface.

8.2. Homologies

Homology is a special kind of interaction between FPs, which seems to be very useful for applications. For example, consider the two FP (chains) $FP_1 = abcdm$ and $FP_2 = acbm$, where each letter denotes also FP so, that FPs, corresponded to similar letters, can bind (recognize) one with the other. Then FP_1 and FP_2 can bind according to their homologies, as shown in Fig.7a or in Fig.7b, where homologous codes are linked by vertical lines. As one can see from Fig.7, such interaction is able to expose similar letters, as well as locations, where similarity violates (*loops*).

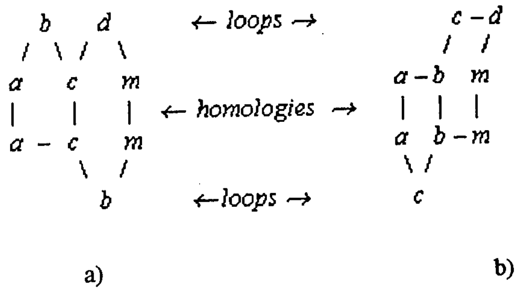


Fig.7. Homologous interactions

As we've shown in [9], formal grammars in general aren't able to describe all the loops of homologous interactions. So we've developed several computational methods of detecting and visualization homologies, as well as their applications to research of proteins and to medical diagnosis [10].
It worth to denote also a model of synchronizing events in computers' networks, based on the developed approach. We've shown [9], that the model needs not to introduce any traditional notion of "time". Nevertheless, the two well known protocols of synchronizing, based on scalar and vector clocks, are treated as a special cases of the model.

8.3. Complex evaluation

The task of complex evaluation states as follows [11]. Let a set of special ecological marks (*indicators*) and/or parameters (SEM) is given. It's required to find out its complex ecological mark, or evaluation (CEE). In other words, it's required to classify this set by assigning it one of the mark (class), denoted, usually, as a numbers *1,2,3,* and so on.

The general solution of the task comprises two stages: learning and recognition. At the stage of learning the set of typical patterns of SEM is being chosen. This can be the results of monitoring for the areas with known CEE, or data, elaborated by experts. Then, using such data, several standard patterns of SEM are being formed for the every mark of CEE. At the stage of recognition given set of SEM is being compared with the standard patterns and the pattern, which is the most "like" the given pattern, determined the sought CEE.
The main differences of developed CEE approach, based on the FIN theory, from the known approaches to pattern recognition can be reduced to the following two features. First, at the frame of the CEE approach, the sets of SEM are coded by parameters of FPs. Second, the degree of similarity between sets of SEM is determined by value of binding energy between corresponding FPs. As a result, the needed value of CEE is determined by that standard FP (antibody), which has the minimal binding energy (the maximal strength of binding) with the given FP.
The developed approach has allowed to solve the following practical tasks:

- processing uncompleted, unprecise and even conflicting data from ecological atlas of St.Petersburg;
- computing the CEE map for ecological atlas of Kaliningrad;
- detecting concrete dependence between quality of environment and morbidity of children for Tula districts;
- detecting similarity in dynamics of infectional morbidity in Russia;
- complex evaluation of population's health for St.Petersburg and the all subjects of Russian Federation.

9. Conclusion

It's said, that there is nothing more practical, than a good theory. It's said also, that any theory is a theory in just the same degree of how much mathematics it contains. From our viewpoint, such obstacles are very important for multi-agent systems, which need a "good theory" in the modern stage of development.
From the other side, we have IS as an excellent example of natural multi-agent system, and we have proteins as its basic agents. Moreover, this approach to multi-agent systems seems to posses a row of advantages comparatively with the AI, because:

- IS is much far studied, than the brain, and molecular immunology develops as a leading direction of the modern science;
- the properties of IS are the straight consequence of the key biomolecular mechanisms of information processing by proteins;
- simulating of the IS begins to give a qualitatively new computational models;
- apparently, properties of proteins seems to be a more promising key to the origin of intellect's mechanisms.

It worth to note, that the theory of FP, as a basic agent of IS, gives also a practical approach to developing a special chips, which we've proposed to call *immunocomputers* [3]. Besides, the properties of IS admit to hope, that immunocomputers would be able to overcome the main deficiencies, which block the wide application of neurocomputers in those fields, where errors cost too much.

Bibliography

1. Artificial Immune Systems and Their Applications (ed. D.Dasgupta).- Springer-Verlag, 1998.
2. Tarakanov A.O. Mathematical Models for Biomolecular Information Processing: Formal Peptide Instead of Formal Neuron // Russian Academy of Sciences, Problems of Informatization, 1998, N1, p.46-51 (in Russian)
3. Tarakanov A.O. Formal Immune Networks: Mathematical Theory and Technology of Artificial intelligence.- In book: Theoretical Basis and Applied Problems of Intelligent Information Technologies (ed. R.M.Yusupov).- St.Petersburg, SPIIRAS, 1998, p.65-70 (in Russian).
4. Teniere L. Basis of Structural Syntaxis.- Moscow, 1988 (in Russian, translated from French).
5. Gorodetski V.I., Tarakanov A.O. Processing planning based on inference in attributive context-free grammars.- In book: Mathematical methods for algorithms synthesis and analysis (eds. Slissenko A.O., Solovyev S.V.).- Leningrad: Nauka, 1990, p.37-48 (in Russian).
6. Tarakanov A.O. Semantic Models for Synthesis Space Navigation Software // Proc. of the 2-d Workshop on the Problems of Applied Space Ballistics.- Moscow, USSR, 1986, p.84-88, 195-198 (in Russian).
7. Tarakanov A.O. Optimization of One Kind Inter-Orbirts Transfer by Theory of Catastrophes // Russian Academy of Sciences, Technical Cybernetics, N2, 1992, p,77-81 (in Russian).
8. DeBoer R.J., Segel L.A., Perelson A.S. Pattern formation in one and two dimensional shape space models of the immune system // J. Theoret. Biol., 1992, v.155, 295-333.
9. Gorodetski V.I., Tarakanov A.O. Genetic Like Model for High Speed Networks // Proc.of First Int. Workshop on High Speed Networks and Open Distributed Platforms (eds.V.Tschammer and M.Smirnov).- St.Petersburg - Berlin, 1995, p.189-193.
10. Tarakanov A.O., Tumanov M.V. Modern Mathematical Methods for Complex Evaluation of Health (ed. R.M.Yusupov).- St.Petersburg, SPIIRAS, 1998 (in Russian)
11. Tarakanov A.O. Complex Ecological Evaluation by Mathematical Model of Molecular Recognition // Proc. of Int. Workshop "Tools for Mathematical Modeling" (MATHTOOLS'97).- St.Petersburg, Russia, 1998, p.62-67.

BID SELECTION IN MULTI-AGENT CONTRACTING: EXPERIMENTAL RESULTS

Maksim Tsvetovat, John Collins, Rashmi Sundareswara,
Joshua van Tonder, Maria Gini
Department of Computer Science and Engineering University of Minnesota
tsvetova, jcollins, sundares, jtonder, gini @cs.umn.edu
Bamshad Mobasher
School of Computer Science, Telecommunications, and Information Systems, DePaul University
mobasher@cs.umn.edu

Abstract

In an automated contracting environment, where a "contractor" agent must negotiate with other self-interested "supplier" agents in order to execute its plans, there is a tradeoff between giving the suppliers sufficient flexibility to incorporate the requirements of the contractor's call-for-bids into their own resource schedules, and ensuring that any bids received can be composed into a feasible plan. In this paper, we introduce a bid evaluation algorithm that incorporates cost, task coverage, temporal feasibility, and risk estimation. Using this evaluation process, we provide an empirical study of the tradeoffs between flexibility, plan feasibility, and cost in the context of our MAGNET multi-agent contracting market infrastructure.

Introduction

The University of Minnesota's MAGNET (Multi-Agent Negotiation Testbed) system is an innovative agent-based approach to complex contracting and supply-chain management problems (Collins *et al.* 1998). It is designed to support the execution of complex plans among a population of independent, autonomous, heterogeneous, self-interested agents. We call this activity *Plan Execution By Contracting.*

The MAGNET system is based on three elements:

1. The *agents.* Each agent is an independent entity, with its own structure, goals, and resources. In general, the resources under "control" of an individual agent are not sufficient to satisfy that agent's goals, and so the agent must negotiate with other agents in its environment in order to meet its goals.
2. The *market.* Agents find each other and carry out negotiations through a distributed infrastructure that enforces protocol rules, limits opportunities for fraud and counterspeculation, and tracks the requests, commitments, and progress toward toward goals among the agent population.
3. The *protocol.* Although the basic structure of the agents and the market will support a variety of negotiation protocols, we have developed a finite, leveled commitment protocol that limits the time and bandwidth required for the negotiation process without limiting the scope of possible agreements.

Because it is based on a market-based economic model, a MAGNET system acts to allocate resources to their highest-value uses over time in a completely distributed fashion. Because MAGNET agents are heterogeneous and self-interested, they may represent real-world entities who may tune their levels of cooperation and competitiveness to suit their own needs.

Plan Execution by Contracting is designed to extend the applicability of agent negotiation to new domains, where schedules for production and delivery affect the cost and feasibility of services and products, and where monitoring the performance of task execution is an essential part of the process. This is specially important for application domains such as logistics, dynamic planning and scheduling, coordination of supply-chain management with production scheduling (Swaminathan, Smith, & Sadeh 1997), where what is important is flexibility, ease of use, quality, performance, as opposed to just cost (Helper 1991).

In general, a MAGNET agent has four basic functions: planning, negotiation, execution monitoring, and resource management. Within the scope of a negotiation, we distinguish between two agent *roles*, the *Contractor* and the *Supplier.* A Contractor is an agent who has a plan to satisfy some goal, and needs resources outside its direct control in order to carry out that plan. The plan may have a *value* that varies over time. In response to a *call-for-bids*, some set of Supplier agents may offer to provide the requested resources or services, for specified prices, over specified time periods. The Contractor agent might specify priorities for tasks and be willing to pay a premium to ensure that higher priority tasks are covered.

Once the Contractor agent receives the bids from Supplier agents, it must evaluate the bids based on cost and time constraints, and select the optimal set of bids (or parts thereof) which can satisfy its goals. The resulting *task assignment* forms the basis of an initial schedule for the execution of the tasks.

In this paper, we introduce a bid evaluation process for automated contracting that incorporates cost, task coverage, temporal feasibility, and risk estimation. Using this evaluation process, we provide an empirical study of the tradeoffs between flexibility, plan feasibility, and cost in the context of the MAGNET market infrastructure. Our experimental evaluation will shed light on the emergent behavior of agents in a multi-agent contracting environment, and in particular, on task allocation strategies for the Contractor agent. The bid evaluation process is of general interest and can be applied in the context of other contracting protocols or combinatorial auctions.

This paper is organized as follows. We describe our study on bid selection in Section 2, and the experimental setup in Section 3. In Section 4 we present our empirical results on tradeoffs between flexibility in plan specification and cost/risk factors in plan execution. Section 5 covers the background and related work.

Agent Interactions in Planning by Contracting

In this section we briefly describe the protocol that supports the Plan Execution by Contracting model. The negotiation portion of the protocol is a finite 3-step process that begins when a Contractor agent initiates a session and issues a Call For Bids.

The Plan Execution by Contracting protocol begins after the session has been initiated by a Contractor agent: the contractor issues a call-for-bids, suppliers reply with bids, and the contractor accepts the bids it chooses with bid-accept messages. Another set of messages, including release, completion, and decommitment, are used to manage the progress of the plan once bids have been awarded. We have avoided the need for open-ended negotiation by means of bid break downs and a time-based decommitment penalty.

1. The Contractor agent starts with a goal to be achieved, along with a (possibly time-varying) value function for the goal. In the simple case, the value of a goal is some fixed amount until a deadline, after which it becomes zero. A more realistic case, for some domains, would see the value of the goal as a decreasing function of time; it is worth achieving only if at the time of achievement its value is higher than the cost of achieving it.
2. The Contractor agent produces a plan to achieve the goal, consisting of a set of tasks and precedence constraints among the tasks. In this process, it uses the domain model from the *Market Ontology.* The domain model includes terms for the products or services within the domain, as well as terminology for quality, quantity, features, terms and conditions of business, etc.
3. The Contractor agent breaks the plan down into one or more discrete bid packages. which among them include all tasks in the plan. The agent then enters the market, either by creating a new session or by re-entering an open session to post a call-for-bids and acquire a set of bids.
4. The market session makes the call-for-bids available to all the appropriate suppliers (those who are registered with the market, and are able to perform the necessary tasks.)
5. Each supplier inspects the call-for-bids, and decides whether or not it should respond with a bid, according to its resources, time constraints, and knowledge of the work to be done. If the supplier chooses to respond, it submits a bid in which it must indicate the cost (to the Contractor), the time window, and the estimated duration of the work. A bid-accept deadline might also be included, as well as a penalty function for each subtask which the Contractor will have to pay if it commits to giving this supplier the work but then decides to decommit.

   Each supplier can send multiple bids for each call-for-bids, each including different costs and time windows, but each supplier will be awarded only one bid combination (or part of one). This is to enable the supplier to send many bids, but not over commit itself. The bid is a commitment by the supplier to do the work listed in the bid, should the Contractor accept it.
6. The Contractor agent selects a set of bids from among the returned bids that form a feasible and approximately minimum-cost assignment of plan elements to resources. In the process of selecting bids, it builds a time map that represents the tasks and their timing relationships, including, for each task, temporal constraints, early start time, late finish time, and duration. The Contractor agent notifies the Supplier agents whose bids have been selected.
7. The Contractor agent, using the time map, monitors execution of the plan. In the process, it receives notifications through the market session of significant events related to each of the awarded bids. Whenever an event occurs that threatens the feasibility or timeliness of plan completion, it takes action to repair or abandon the plan.

The main goal of our experiments is to observe the relationships among numbers of suppliers, the temporal flexibility offered to suppliers, and the price and risk factors experienced by a customer agent.

The experiments are based on the following assumptions:

- different suppliers will require varying amounts of time to perform a given task;
- suppliers are motivated to maximize the profitable utilization of the resources under their control;
- suppliers may or may not have flexibility in resource loading over time. For a supplier with a fixed set of resources, bids can only be submitted for time intervals where resources are uncommitted.

We expect that larger numbers of bidders will result in lower prices, a higher probability of being able to compose a feasible plan, and lower risk factors. We also expect that giving bidders larger time windows in which to bid will result in larger numbers of bids and lower costs, but may also result in greater difficulty finding feasible compositions of bids. If the number of bidders is large enough, larger time windows should allow the customer to compose lower-risk plans by finding bid combinations that contain more slack time for tasks that are likely to involve higher risks, such as cases where only one bidder is available for a given task.

Experimental Setup

The experimental setup includes three main components: a MAGNET Server as described in (Collins *et al.* 1998), a Contractor Agent that generates plans, requests bids, and evaluates bids, and a Supplier Agent that generates and submits bids.

Contractor Agent: Generate Plan and Issue Call for Bids

The Contractor agent generates a plan

$$\mathcal{P} = \{\mathcal{S}, \mathcal{R}\}$$

where $\mathcal{S}$ is a set of tasks, and $\mathcal{R}$ is a set of precedence relations $\mathcal{R}_f = \{f' \in \mathcal{S} \,|\, f' \prec f\}$, the set of other tasks $s' \in \mathcal{S}$ whose completion must precede the start of s.

The call-for-bids C contains a subset of the tasks in the plan P with their precedence relations. Let $C.t_{es}(s)$ denote the early start time for task s as specified in a call-for-bids C. The notation $s' \prec s$ denotes the constraint $t_f(s') \leq t_s(s)$, meaning that task s' must be complete before task s can start. A necessary condition for this is $(t_{es}(s') + d(s')) \leq (t_{lf}(s) - d(s))$, where $d(s)$ refers to the duration of task s.

For each task s in the call-for-bids C, the following variables must be specified:

- a time window, consisting of an earliest start time $C.t_{es}(s)$ and a latest finish time $C.t_{lf}(s)$,
- a set of precedence relationships $C.Pred(s) = \{s' \in C \,|\, s' \prec s\}$, the set of other tasks $s' \in C$ whose completion must precede the start of s, and
- a decommitment penalty $dcom(s)$. This is the penalty the supplier has to pay to the contractor agent if the supplier decommits. (See (Collins *et al.* 1997) for further explanation of the decommitment penalties.)

The agent may have general knowledge of normal durations of tasks, or it may not. In order to minimize bid prices, vendors need flexibility in making resource commitments. Two possible strategies are (a) to use time windows that start at time t_0 and end at time t_{goal}, where t_0 is the start time for the entire plan and t_{goal} is the deadline for achieving the top level goal, and (b) to schedule tasks ahead of time using approximate durations, computing early start and late finish times using the Critical Path algorithm (Hillier & Lieberman 1990). The minimum duration of the entire plan is called the ***makespan*** of the plan. The difference between t_{goal} and the latest early finish time is called the ***total slack*** of the plan. If t_{goal} is set equal to $t_0 + makespan$, then the total slack is 0, and all tasks for which $t_{ef} = t_{lf}$ are called *critical* tasks. Paths in the graph through critical tasks are called *critical paths.*

For the purpose of this experiment, the plans consist of identical tasks. Each task has an expected duration $d_e(s)$ that we arbitrarily set to 1 hour. In the full MAGNET system, expected task durations will be defined by the market ontology.

In order to allow for variability in actual task durations, and to allow suppliers some degree of flexibility in their resource scheduling, the specified start and finish times are relaxed from their expected values by varying degrees. These times are computed as follows:

- *start time*: The target start time t_0 is set to an arbitrary point in time in the near future.
- *deadline*: Using expected task durations $d_e(s)$, the earliest possible completion time for the entire set of tasks is found by analyzing the dependency graph. The target completion time t_d is then set to allow an overall slack of 10% to account for some uncertainty on the supplier side.
- *time windows*: Early start and late finish time specifications are computed for each task $s \in S$. For each root task which has no predecessors, $C.t_{es}(s) = t_0$. For each leaf task which has no successors, $C.t_{lf}(s) = t_d$. For all other tasks,

  $$C.t_{es}(s) = \max_{s' \in pred(s)} (C.t_{es}(s') + d_a(s'))$$

  and

  $$C.t_{lf}(s) = \min_{s' \in succ(s)} (C.t_{lf}(s') - d_a(s')),$$

  where d_a is an adjusted duration used to introduce flexibility into the specification and allow for variation in actual task durations. The ratio $\frac{d_e(s)}{d_a(s)}$ is one of the experimental variables, as described below.

Supplier Agent: Generate Bids The supplier agent is a test agent that masquerades as an entire community of suppliers. Each time a new Call For Bids is announced by the Market, the supplier attempts to generate some number of bids.

As described in (Jamison 1997), each bid represents an offer to execute some subset of the tasks specified in the Call For Bids. A price for the whole set is specified, and for each task in the set, a price, early start time, late finish time, and maximum duration are specified. It is a requirement of the protocol that the time parameters in a bid are within the time windows specified in the Call For Bids. For our experiments, bids are generated randomly as follows:

- *select a set of tasks*: A contiguous set of tasks $b.S$ is selected by choosing an initial task and then randomly following predecessor and successor links. Bids consisting of contiguous tasks are realistic for many domains, such as manufacturing.
- *set duration*: The duration $d(s)$ for each task is a normally distributed random variable with a mean of 1 hour and standard deviation of 10
- *determine feasibility*: If the duration is longer than the time window

$$[C.t_{es}(s)\,,\, C.t_{lf}(s)]$$

specified in the Call For Bids, the task is dropped.
- *fill in bid data*: The following steps are performed on each selected task $s \in b.S$:
  - *determine bid start and finish times*: The expected early start time of the task $t_{es}(s)$ is selected as a uniformly distributed variable between the early start time $C.t_{es}(s)$ for s specified in the Call For Bids, and a late start time derived by subtracting $d(s)$ from the late finish time $C.t_{lf}(s)$ specified in the Call For Bids. The resulting bid early start time $b.t_{es}(s)$ is inserted in the Bid. A bid late finish time $b.t_{lf}(s)$ is similarly determined by picking a random value in the interval

$$[b.t_{es}(s) + d(s)\,,\, C.t_{lf}(s)]$$

  - *determine price*: Price $b.p(s)$ for a task s is set as

$$b.p(s) = \rho \times C.flex(s) \times b.commit(s)$$

,where ρ is the initial price. Due to the fact that all tasks in this experiment are identical, the situation is similar to commodity markets where goods are sold at similar prices by all suppliers. *flex* is a flexibility discount, representing the fact that the cost of production is often lower when the supplier has some temporal flexibility in scheduling the work. *commit* is a resource commitment premium, representing the extra cost in keep production resources committed while waiting for the Contractor to release the job.

For the purpose of this experiment, ρ, or initial price, is a normally-distributed value with a mean of 1000 and a standard deviation of 2%. The value of *flex* varies linearly from 1.0 to 0.8 as the ratio of the time window in the call for bids to the task duration ($\frac{C.t_{lf}(s)-C.t_{es}(s)}{d(s)}$) increases from 1.0 to 2.0, and then remains at 0.8. Similarly, the value of *commit* varies linearly from 1.0 to 1.2 as the ratio of the time window in the bid to the task duration ($\frac{b.t_{lf}(s)-b.t_{es}(s)}{d(s)}$) increases from 1.0 to 2.0, and then remains at 1.2. All these values are arbitrarily chosen to create a reasonable spread of prices and time windows in the bids.

  - *set decommit penalty*: The decommitment penalty that the Contractor must pay for backing out of a bid is chosen as a random value between 0 and the price of the bid.
- *set overall discount*: If the supplier is bidding on multiple contiguous tasks, it may offer a discount in exchange for acceptance of the entire combination. The overall price for the bid is

$$b.p = \sum_{s \in b.S} (b.p(s) \times b.discount)$$

where *b.discount* is a random value between 1.0 and 0.8 (corresponding to the overall discount of 0%-20%).

Contractor Agent: Evaluate Bids Once the bidding deadline is past, the Contractor evaluates the set of bids in an attempt to find a combination that minimizes a combination of cost and risk, provides coverage of all tasks, and allows for a feasible schedule. For each bid, the Contractor has the option of selecting the entire bid and paying the overall discounted price $b.p$, or selecting a subset of the task bids from $b.S$.

Let B be the set of returned bids, each bid $b \in B$ includes a price $b.price(S')$ for some subset S' of the elements of the call-for-bids, prices $b.price(s)$ for individual elements of the bid subset $s \in S'$, and timing information for the bid elements. When $b.price(S') \leq \sum_{s \in S'} b.price(s)$ then a *discount* is associated with the bid b. Timing information for a bid b and a task s includes early start $b.t_{es}(s)$, late finish $b.t_{lf}(s)$, and duration $b.d(s)$ for each task in the bid. The semantics of a bid is that a supplier is willing to perform the task s for the bid price $b.price(s)$ starting at any time $t_s(s)$ such that $b.t_{es}(s) \leq t_s(s) \leq (b.t_{lf}(s) - b.d(s))$, and finishing at time $(t_s(s) + b.d(s))$.

The evaluator uses a stochastic search, executing the following steps:

- *choose a node*: A previous partial or complete solution is chosen as a base for the next expansion. A node is a partial or complete mapping of tasks to bids. The root node is the mapping of all tasks to no bids. The nodes are kept in an ordered queue, and the choice is made by choosing an exponentially-distributed value whose mean is 10% of the range of evaluations in the queue. The range is limited; nodes with evaluations that fall off the end are dropped, and when a new best node is found, the tail is trimmed.
- *choose an expansion*: Expansions are made by adding a bid or individual task-bid to a node. The probability of adding a task-bid rather than a bid is equal to the current coverage factor of the node being expanded, where the coverage factor is the proportion of tasks that are mapped to bids. This is to encourage use of discounted whole bids when coverage is low, and individual task-bids when coverage is high and the addition of a bid is likely to

displace other bids from the mapping because of task overlaps. Bids that have already been used to expand a given node are disallowed, as are bids that are in the *tabu list*, which contains the last 10 expansions in the parentage of the node. In addition, if there are any "singleton" bids, or tasks for which only one bid was submitted, bids for those tasks may not be displaced by expansions.

- *evaluate the resulting node*: If a valid node was produced by the expansion, it is tested for coverage, feasibility, cost, and risk, as follows:
  - *coverage*: A penalty is added for each task that is not mapped to a bid.
  - *feasibility*: The Critical Path algorithm is run across the task-bid mapping, with unmapped nodes assumed to have a duration of 0. A penalty proportional to the total negative slack is added.
  - *cost*: The total cost of all mapped bids is added.
  - *risk*: In general, risk factors include decommitment cost, recovery cost, loss of value as the end date is delayed, cost of plan failure, and other factors. For the purposes of this experiment, we have chosen to focus on the expected cost of decommitment penalties. These are penalties that must be paid by the Contractor to suppliers if the Contractor backs out of a commitment. We assume that the Contractor will only do this with regard to a particular task if execution of a prior task experiences an unrecoverable failure. The total expected decommitment cost is

$$c_d(total) = \sum_{s \in S}(P_{pud}(s) \times c_d(s))$$

where $P_{pud}(s)$ is the probability that a task $s' \in pred^*(s)$ in the transitive closure of the predecessors of s will experience an unrecoverable decommitment. We calculate $P_{pud}(s)$ as

$$P_{pud}(s) = 1 - \prod_{s' \in pred^*(s)} (1 - (P_{sd}(s') \times (1 - P_{rec}(s'))))$$

where $P_{sd}(s')$ is the probability that the supplier chosen for task s' will decommit, and $P_{rec}(s')$ is the probability that recovery can be achieved after decommitment:

$$P_{rec}(s) = (1 - \frac{1}{1 + \frac{A.t_{lf}(s) - A.t_{es}(s)}{b.d(s)}})(1 - \frac{1}{bidCount(s)})$$

The justification for this formula is that the probability of recovery increases with increasing slack, and with a larger number of available suppliers as evidenced by the number of bids received for the task.

- *stopping criterion*: The search ends when a number of expansions equal to 20 times the number of bids have been attempted without finding an improvement. The best feasible and covered solution, if any, is returned. The number 20 is arbitrarily chosen, in our experiments we have found out that this prevents continuing the search when no progress is made but, at the same time, does not terminate the search too soon.

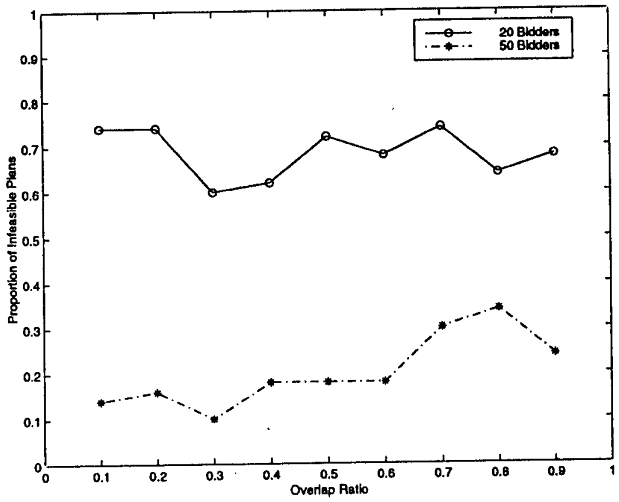


Figure 1: Infeasibility vs. Slack

Experimental Results

We explored two dimensions of the bid-evaluation problem with the setup described above. The first dimension is simply the number of bidders. One set was run with 20 bidders to simulate a rather small set of possible suppliers. We ignore the possible effects of collusion and collaboration among vendors in a restricted market. The other set included 50 bidders.

For each of the two sets, we generated Calls For Bids with 10 identical tasks, and time windows varied $0.1 \le \frac{d_e(s)}{d_a(s)} \le 0.9$. We ran 50 iterations for each Call For Bids, each iteration consisting of the following steps:

- *Contractor Agent*: compose and submit call for bids
- *Supplier Agent*: compose and send out bids
- *Contractor Agent*: evaluate bids and output the best plan

As we can see from Figure 1, the probability of finding a feasible, covered plan is reduced dramatically with small numbers of bidders. When time windows are short, the infeasibility test in the bid preparation process results in smaller numbers of bids, with the bids received being composed of fewer tasks. However when the time windows are increased, the effect of larger number of (larger) bids is more than offset by fewer combinations that form temporally feasible plans.

Figure 2 shows the relation between expected decommitment cost and time-window size. Given that the deadline for the overall plan is constant across these runs, and given that the average task duration is also constant, it is interesting that the variation of

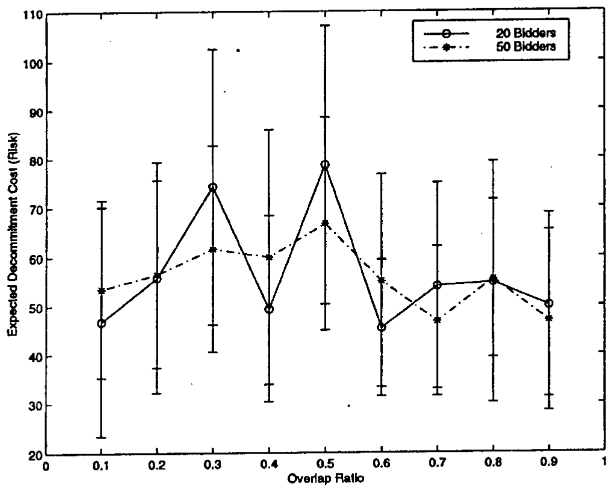


Figure 2: Risk vs. Slack

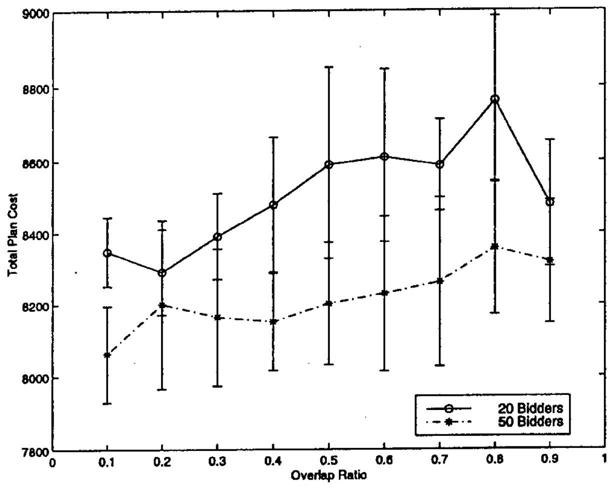


Figure 3: Cost vs. Slack

the risk evaluation decreases for the larger number of bidders. This signifies the fact that it is easier to recover from a partial failure in markets with a large number of suppliers.

With the larger number of bidders there is a relationship between the size of the time windows specified in the Call For Bids and the expected decommitment cost. This is because the bidders are generally composing bids with larger time windows, and even though many of them cannot be composed while preserving precedence relations, the evaluator is able to find combinations that maximize slack early in the plan, where it has the greatest impact on expected decommitment cost.

Figure 3 shows the expected relationship between cost and number of bidders. The increasing cost with larger time windows is due to the same effect that reduces risk: suppliers are specifying larger time windows in the bids, and are applying the resource-commitment premium. In a more realistic situation, bidders could submit both a higher-cost, more flexible and lower-risk bid, along with a lower-cost, less flexible and higher-risk bid, allowing the customer to make the decision.

Related Work

Markets play an essential role in the economy, by facilitating the exchange of information, goods, and services (Bakos 1998), and there is growing evidence that software agents will play an increasing role as mediators in electronic markets (Guttman, Moukas, & Maes 1998; Sycara, Decker, & Williamson 1997). Mediators typically provide services such as searching for a product or supplier, negotiating the terms of a deal, providing payment services, and ensuring delivery of goods.

There is a growing interest in using market and economics principles to model the interactions of multiple agents (Mullen & Wellman 1996). Many market-based architectures have been proposed for multiple agents (see, for instance, (Rodriguez *et al.* 1997; Mullen & Wellman 1996)). KQML is emerging as the preferred communication language for agents (Finin, Labrou, & Mayfield 1997), but other languages have been proposed for electronic commerce, such as CommerceNet's eCo/CBL (Tenenbaum, Chowdhry, & Hughes 1997).

Of the essential functions of a market identified by (Bakos 1998), (i) matching buyers and sellers, (ii) facilitating transactions, and (iii) providing institutional infrastructure existing software agents mostly satisfy the need to search for product and price information (see, for instance, (Doorenbos, Etzioni, & Weld 1997), but there is a growing need for agents capable of sophisticated automated negotiations (Beam & Segev 1997; Guttman & Maes 1998).

Automated contracting protocols generally assume direct agent-to-agent negotiation. For example, Smith (Smith 1980) pioneered research in communication among cooperating distributed agents with the Contract Net protocol. The Contract Net has been extended by Sandholm and Lesser (Sandholm & Lesser 1995) to self-interested agents. In these systems, agents communicate and negotiate directly with each other. In our proposed work, agents interact with each other through a market.

To the extent that we require the existence of an external market mechanism as an intermediary, our proposed framework is similar to that of Wellman's market-oriented programming used in AuctionBot (Wurman, Wellman, & Walsh 1998) and The University of Michigan Digital Library. AuctionBot, for instance, supports a variety of auction types each imposing a set of market rules on the agent interactions. In contrast, our framework provides explicit market mechanisms which can not only specify and enforce auction parameters, but also support more complex automated contracting.

Rosenschein and Zlotkin (Rosenschein & Zlotkin 1994) showed how the behavior of the agents can be influenced by the set of rules that the system designers choose for the agents' environment. In their study the agents are homogeneous and there are no side payments. In other words, the goal is to share the work, not to pay for work. Sandholm's agents (Sandholm & Lesser 1995; Sandholm 1996) redistribute work among themselves by a contracting mechanism. Unlike Rosenschein and Zlotkin, Sandholm considers agreements involving explicit payments, but he also assumes that the agents are homogeneous – they have equivalent capabilities, and any agent can handle any task. We do not make any of these assumptions. Our agents are heterogeneous, and decide on their own what tasks to handle by responding to a call for bids that requires specific tasks or products within a specified time window.

Conclusions and Future Work

We have presented a bid evaluation process for automated contracting that incorporates cost, task coverage, temporal feasibility, and risk estimation, and we have provided, using this evaluation process, an empirical study of the tradeoffs between flexibility, plan feasibility, and cost in the context of the MAGNET market infrastructure. We show that a Contractor agent can make tradeoffs between price, feasibility, and risk by adjusting the temporal specifications of the tasks to be performed. We also show that the nature of these tradeoffs varies with the number of potential suppliers in the market.

This/work raises several interesting questions for future research. A game-theoretic analysis of the protocol could be done to determine optimal strategies for its use by agents. In particular, how should the time windows and decommitment penalties be used, and how should proposed decommitment functions be evaluated when computing marginal costs of plan alternatives. The risk evaluation process also needs to be expanded to include recovery cost estimates and time-varying decommitment penalties.

In cases where the value of the goal is a decreasing function of time, or where schedule slack is low and probability of missing the goal deadline is high, there may be value in offering to suppliers not only a penalty for decommitment or failure to perform, but also a premium for early completion of subtasks. We will study how to calculate the value of such a premium, and develop methods for incorporating this into the negotiation protocol.

References

Bakos, Y. 1998. The emerging role of electronic marketplaces on the Internet. *Comm. of the ACM* 41(8):33–42.

Beam, C., and Segev, A. 1997. Automated negotiations: A survey of the state of the art. Technical Report CITM Working Paper 96-WP-1022, Walter A. Haas School of Business.

Collins, J.; Jamison, S.; Gini, M.; and Mobasher, B. 1997. Temporal strategies in a multi-agent contracting protocol. In *AAAI-97 Workshop on AI in Electronic Commerce.*

Collins, J.; Youngdahl, B.; Jamison, S.; Mobasher, B.; and Gini, M. 1998. A market architecture for multi-agent contracting. In *Proceedings of the Second International Conference on Autonomous Agents*, 285–292.

Doorenbos, B.; Etzioni, O.; and Weld, D. 1997. A scalable comparison-shopping agent for the world-wide web. In *Proceedings of the First International Conference on Autonomous Agents*, 39–48.

Finin, T.; Labrou, Y.; and Mayfield, J. 1997. Kqml as an agent communication language. In Bradshaw, J., ed., *Software Agents.* Cambridge, MA: MIT Press. 291–316.

Guttman, R. H., and Maes, P. 1998. Cooperative vs competitive multi-agent negotiations in retail electronic commerce. In *2nd Int'l Workshop on Cooperative Information Agents (CIA '98).*

Guttman, R. H.; Moukas, A. G.; and Maes, P. 1998. Agent-mediated electronic commerce: a survey. *Knowledge Engineering Review.*

Helper, S. 1991. How much has really changed between us manufacturers and their suppliers. *Sloan Management Review* 32(4):15–28.

Hillier, F. S., and Lieberman, G. J. 1990. *Introduction to Operations Research.* McGraw-Hill.

Jamison, S. 1997. A negotiation protocol and market agent model for complex transactions in electronic commerce. Technical report, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, MN.

Mullen, T., and Wellman, M. 1996. Some issues in the design of market-oriented agents. In Wooldridge, M.; Mueller, J.; and Tambe, M., eds., *Intelligent Agents: Theories, Architectures, and Languages*, volume II. Springer-Verlag.

Rodriguez, J. A.; Noriega, F.; Sierra, C.; and Padget, J. 1997. FM96.5 - a Java-based electronic auction house. In *Second Int'l Conf on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97).*

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter.* Cambridge, MA: MIT Press.

Sandholm, T., and Lesser, V. 1995. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *1st International Conf. on Multiagent Systems*, 328–335.

Sandholm, T. W. 1996. *Negotiation Among Self-Interested Computationally Limited Agents.* Ph.D. Dissertation, University of Massachusetts.

Smith, R. G. 1980. The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Trans. on Computers* 29(12):1104–1113.

Swaminathan, J. M.; Smith, S.; and Sadeh, N. 1997. Modeling the dynamics of supply chains: A multi-agent approach. *Decision Sciences.*

Sycara, K.; Decker, K.; and Williamson, M. 1997. Middle-agents for the Internet. In *Proceedings of the 15th Joint Conference on Artificial Intelligence.*

Tennenbaum, J. M.; Chowdhry, T. S.; and Hughes, K. 1997. eCo System: CommerceNet's architectural framework for internet commerce. Technical report, Object Management Group, Cambridge, MA.

Wurman, P.; Wellman, M.; and Walsh, W. 1998. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Second Int'l Conf. on Autonomous Agents.*

ADJUSTABLE AUTONOMY, DELEGATION AND DISTRIBUTION OF DECISION MAKING

Harko Verhagen and Johan Kummeneje

The DECIDE Research Group
Department of Computer and Systems Sciences
The Royal Institute of Technology and Stockholm University
Electrum 230, SE-16440 Kista, Sweden
verhagen, johank@dsv.su.se

Abstract

In this paper we will illustrate the possible uses of norms in multi-agent decision making. Norms as obligations serve to generate action repertoires and norms as constraints on behavior enable pronouncers to filter out subrational decisions. These are combined with adjustable autonomy to produce highly adaptive agents. The theoretical framework is tested in simulations of norm-spreading and robotic soccer.

Keywords: *multi-agent systems, norms, robotic soccer, learning, pronouncers*

1. Introduction

The focus of this paper is on how agents may obtain a balance between following the norms of the coalition and individualistic decision making (i.e., maximizing personal profit). Since agents may act in dynamic environments, the choice between these two extremes should not be fixed beforehand. In other words, agents should be able to adjust their autonomy with respect to the coalition at run time. Another interesting issue is how the set of shared norms comes about.
Adjustable autonomy is a recently introduced term from the field of autonomous systems in space exploration [11]. The general idea is that autonomous systems adapt the level of autonomy to the situation at hand. One aspect of agent autonomy is largely ignored in [11], viz. the level of norms. Designing norm-regulated or norm-autonomous agents will enhance the possibilities of accurate models of humans for the agents and vice versa. The need for override of human commands by the artificial agents can be modeled more clearly when using the concept of norms.
Furthermore, norms can be used by external counseling devices such as pronouncers [4]. Agents finding themselves in a situations too complex for them to make a rational decision may ask an external entity for advice. A special case of such an entity is a pronouncer. A pronouncer is an artificial decision maker which itself is not an agent nor part of the coalition. Pronouncers take as input the decision tree of the agent, a description of the situation and the set of norms of the agent and has as an output the most rational choice given the information provided. The use of pronouncers for advice will reduce the dependence of agents on other agents (human or artificial). It will also enable the artificial and human agents together forming a society to be less dependent on any outside control.
Individual agents participate in several coalitions and may revise their standing towards the coalition and its members during execution. This enhances their self-organizing capabilities thus increasing their flexibility and (re)action repertoire.
First we will look more closely at the concept of autonomy before discussion ways of delegation and distribution of decision making. After this we will introduce the two simulation models developed to test the usability of these concepts in multi-agent systems and discuss the results obtained so far. Finally we will indicate possible topics for further research.

2. Autonomy

Autonomy is relative to something external to the agent; an agent is not autonomous in an abstract sense but is autonomous on some level with respect to another entity, be it the environment, other agents, or even the developers of the agent. This point is also made in [7] where it is stated that an agent's autonomy is necessarily limited since it is situated. Unlimited autonomy renders an agent solipsistic since the agent does not allow for any input from its environment. Autonomy can be viewed in at least two ways:

- levels of autonomy as abstraction levels or the control the agent has over its behavior and decision making process
- level of autonomy as level of independence of coalition

The autonomy models developed in multi-agent systems research (e.g., [6], [8], [10] and [24]) are based on the first view on autonomy. These autonomy models are summarized and extended in [23]. In short, decision making takes place at four separate yet connected levels:

- the level of actions
- the level of plans
- the level of goals
- the level of norms

The autonomy model developed in the work on adjustable autonomy on the other hand is focused on the second view of autonomy and describes the ways in which agents may be independent of their coalition members.

3. Delegation and distribution of decision making

Norms can be viewed in different ways. For example, [8] (pp. 91-92) distinguishes norms as:

- constraints on behavior
- ends (or goals)
- obligations

In [3] norms are seen as constraints on behavior. In the current work we view norms as obligations. Viewing norms as obligations includes norms as descriptions of how to behave in a certain situation as well as a description of the distribution of problem solving (via roles). Norms as behavior descriptions can be seen as decision modules that can be used in less complex situations where agents follow the norms of the coalition when making a decision. Based on their knowledge of the norms, agents can expect and thus predict the behavior of fellow coalition members. The function of roles in the normative action model [15] can be seen as an implicit distribution of decision making. The complement to distribution of decision making via norms is the delegation of decision making, where a decision is explicitly designated to a specific agent. A pronouncer is a special case of this since it is not agent nor part of the coalition.

4. Simulation of norm-spreading and norm-internalizing

The simulation model consists of several agents roaming a two dimensional space. The agents form a coalition with one of the agents acting as the leader. Every spot in the two dimensional space may contain either nothing, one piece of resource one, one piece of resource two or one piece of both resources. The agent has a choice to either do nothing, move to another spot or take resource one or resource two (if available). Combining the amount of content alternative with the choice alternatives and outcome alternatives (if the chosen alternative is realized or not) gives 20 combination alternatives in total.

4.1. Description of decision making model

Every agent has a private utility base containing utility values for each of these alternatives (self-model) and a coalition utility base containing the utilities the agent presumes the coalition has for each of the alternatives (coalition model). The coalition model expresses the agent's interpretation of the norms the coalition holds. The degree of autonomy of an agent relative to the coalition determines to what extent the coalition model is followed when making a decision. E.g., an autonomy of 0.4 expresses that in 40 percent of the cases the coalition model is followed instead of the self-model to make a decision.

Choosing an alternative does not mean this is also bound to happen, chance and the agent's skills influence the outcome. An agent updates its self-model and coalition model based on the outcome and the result of the feedback coalition members give in answer to the agent's message containing the chosen alternative.

The influence of the leader of the coalition may also vary. The leadership value expresses the weight of the information provided by the coalition leader. E.g., a leadership value of 2 expresses that the feedback from the leader is twice as important as the feedback of the other coalition members.

4.2. Simulation setups

The following simulations were run: autonomy (on a scale from 0 to 1) had a value of 0.0, 0.4 or 0.8 and the leadership value was 1, 2 or 5. In total this gives 9 simulation setups.

The following two hypotheses were formulated:

Hypothesis 1: the higher the degree of autonomy, the higher the variance of behavior will be.

Hypothesis 2: the higher the leadership value, the lower the variance of behavior will be.

The variance of behavior can be measured in several ways. One way is by determining the difference between an agent's own utility base and the coalition utility base it has. This expresses the norm-internalizing. The mean value over the agents of the mean value of the absolute difference between self-model's utility and group model's utility per choice alternative is the measure used for expressing the norm-internalizing factor of the MAS as a whole. Another measure is the differences in the coalition utility bases over the agents. This expresses the norm-spreading. The spreading of norms is graphically displayed as the mean value of the standard deviation per alternative of the coalition utility of that alternative for each of the agents.

4.3. Implementation of the simulation model

The simulation model is implemented in Java. Each agent is a separate thread. The agents communicate with the environment and each other through a router programmed using JATLite. Varying the settings for the agents requires editing of some datafiles and Java code files and compiling these. All simulation runs were run for 100 minutes. During the simulation run a log file is kept for each agent which gets updated every minute with the agents self-model and coalition model at that point in time.

4.4. Simulation results

The results of some of the simulations are shown in the figures 1 to 4. Figure 1 shows the behavior of the norm-internalizing factor in time with an autonomy of .4 for the three different leadership values.

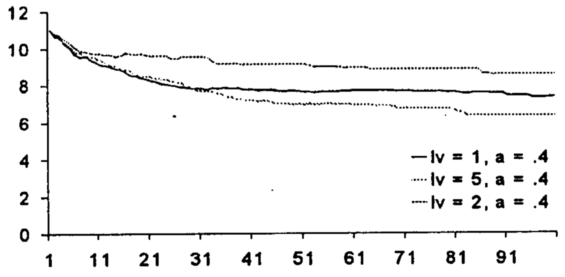


Figure 1: norm-internalizing with autonomy = .4

Figure 2 shows the influence on the norm-internalizing factor of varying the autonomy factor while the leadership value is held constant at 2.

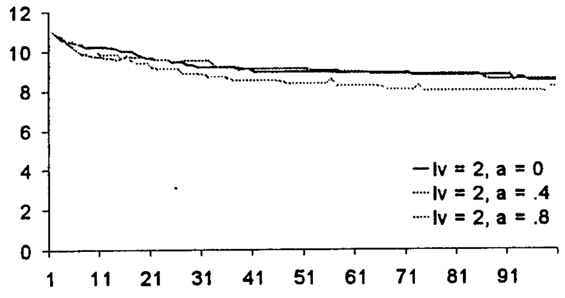


Figure 2: norm-internalizing with leadership = 2

Figure 3 shows how the norm-spreading factor behaves in time for the various leadership value settings while the autonomy factor is set to .4.

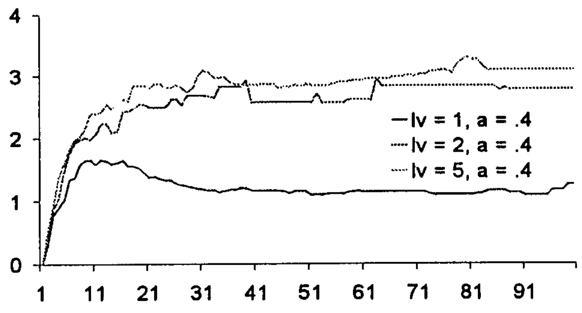


Figure 3: norm-spreading factor with autonomy = .4

Finally, figure 4 depicts the norm-spreading factor over time for a leadership value of 2 and varying the autonomy factor.

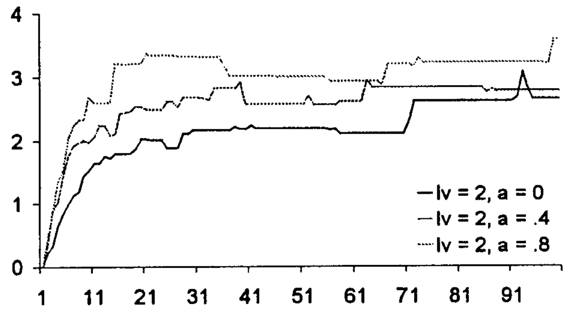


Figure 4: norm-spreading factor with leadership = 2

4.5. Interpretation of the preliminary results

The norm-sharing factor as depicted in figures 3 and 4 complies with both formulated hypotheses. One may observe that increasing the autonomy causes the agents to need more time to reach the maximum value of the norm-sharing factor and it also takes more time for the agents to enter a stable situation. Increasing the leadership value on the other hand makes the agents reach the norm-sharing value's maximum sooner and also get to a stable situation sooner.

The norm-internalizing factor as depicted in figure 1 and figure 2 does not comply with the formulated hypothesis. Several factors may play a role here. One possible cause is that not all situations occur during the simulation. Since the norm bases are only updated for the situations that occur, some utilities do not change during the entire simulation. A second explanation may be that the variance between different runs with the same setting could be greater then the difference between runs with different settings. More elaborate simulations have showed that this may explain for some variation but that the relative positions of the graphs are constant.

5. Robocup

Originating as a standard problem for AI [17], the RoboCup has evolved into a divers field of research with approximately 1500 researchers world-wide in January 1999. The research spans from robotics to social sciences as the formulation of the RoboCup Challenge includes both robotic soccer tournaments and computer simulated soccer tournaments in which research focusing on among others collaborative and individual planning [2] can be conducted. The choice of robotic soccer as a test-bed is quite natural because of the vast range of research items, the limited set of rules, the limited field size, and the unpredictability of a near real-world application. We have chosen to extend the research described in the previous chapter on simulation of norm spreading and norm-internalization, into the field of RoboCup, in order to test the validity of the results found in the previous chapter in a dynamic environment.

5.1. UBU

Utility Based Reasoning Under Uncertainty (UBU) is the result of a research effort in the simulator league of RoboCup. Previous versions have competed both in the World Cup 1998 in Paris [1], and in the Pacific Rim 1998 in Singapore [4]. Our next version which will compete in the World Cup 1999 in Stockholm will have pronouncer calls implemented for the coach agent. Training matches between the old team and the team under development will be played to measure the progress made.

5.2. Team Constellation

The team consists of twelve autonomous processes that represent the eleven players and the coach. Since the players make decisions at a rate of 10 decisions per second and pronouncers usually need more than a second to return their recommendations [25], we do not intend to use pronouncers in the players. Since the coach agent does not participate actively in the game, it is not under the same hard performance requirements. Thus we can wrap a pronouncer into the coach agent.

The RoboCup-environment is very suitable to perform research on autonomy as the team consists of twelve autonomous processes. One of our research topics is to decide what levels of autonomy are satisfactory in order to optimize team-behavior in which situations? We will also research the area of delegation and distribution of decision making by implementing a decision hierarchy between the agents. Depicting the decision-hierarchy on the levels of decision-making as described in section 2, gives the following. The coach will be setting the goals. The group-leaders will decide upon the plans on how to achieve the goals and the individual agents will decide which exact actions have to be performed to realize the plans.

In the development of the team, we will use an approach similar to CMUnited'97 and CMUnited'98 [20], with position based playing. The actual positioning of the players on the field is not interesting at this stage, as we are more interested to see their roles in decision-making. However to reach the highest and toughest level of decision-making we may need to rethink the hierarchy approach and try some form of contract net where the agents will negotiate who is deciding what and setting the norms.

One approach is to use a locker-room-agreement [19]. This approach may be efficient in order to set general norms, but each of the specific cases encountered in the game might need some kind of negotiating. If the negotiating becomes too extensive and time-consuming, it might be better to execute the best possible alternative found in the time available, i.e. just-in-time-algorithm. Our research will be among others focused on where and how to make the decisions on the different levels of decision-making, as well as how norms are adapted and shared within the team during and between games.

5.3. Robocup expected results

We expect to see results similar to the interpreted preliminary results of section 4.5, and also to get a good picture of how decisions are made efficiently in a real-time and dynamic environment, which we hope might be of use in human decision processes. During the experiments with the team, we will probably encounter several ambiguities. This would lead us to changing the number of parameters in the simulations and test our theories in other less complex simulation-models in order to test their validity.

6. Related work

An example of research on obtaining a balance between deliberations at macro and micro levels is [16] in which a proposal for social rationality is developed. The authors try to give a general decision rule based on payoff functions (in terms of benefits and losses) for individuals and society. This group decision rule is however not resistant to individual differences between the agents (i.e., all agents are supposed to follow the group rule) and agents can be member of only one group (i.e., the whole agent systems forms one social system which has no subsystems).

As mentioned previously, the macro-micro link problem and the function of norms is discussed in [8]. In more recent work [9] a model for norm acceptance is developed. In a sense this work is complementary to our work. We do not study the acceptance of norms but presuppose that coalition membership in itself is reason to accept norms. On the other hand, unlike [9] we do not presuppose that accepting norms implies following them.

In a series of publications (c.f., [18]), Tennenholtz has written on the use of social laws as coordination mechanisms. This line of research differs from ours in several aspects. Tennenholtz' social laws are aimed at optimizing the behavior of the agents sharing an environment, the social laws are engineered off-line (and preferably by the designer of the agent system), are used to solve coordination problems and are hard, that is, the agents are not allowed to **not** follow the social laws. In our view, the norms are not necessarily related to optimizing the system, norms preferably emerge from interaction with both the environment and other agents, can be used to solve not only coordination problems but also serve to enable delegation and distribution of decision making, and finally agents may choose to not follow the norms.

In the work on Sugarscape [12] some form of norm spreading is also studied. The model of cultural transmission is too basic and its transmission rule too general to be considered a model of norm spreading reconcilable with deliberating agents.

Peter Stone, who recently received his Ph.D., at the Carnegie Mellon University (CMU), has in co-operation with professor Manuela Veloso and Patrick Riley developed a team, CMUnited, which became the champion of the World Cup 1998. Stone's research is more focused on machine learning [19] than on the social aspects of RoboCup. Milind Tambe and Gal Kaminka of the University of Southern California, have used a rule-based framework approach when developing their team ISIS [21] in Soar. Tambe and Kaminka do research in the field of teamwork and social diagnosis, which is not far from the research on social norms performed in our lab [5] and described in this paper.

7. Discussion and future research

Further simulation experiments will be conducted to draw conclusions as to why the norm-internalizing factor does not comply with the formulated hypothesis. Other future work in the simulation of norm-spreading will include comparison of different initial situations (i.e., a default coalition norm base versus an initial coalition norm base being equal to the self norm base). Another topic for future research will be the formation of coalitions. Previous work on this topic has been focused on game theory and thus individualistic decision making. Inspiration from social theory (e.g., [13], [14], [22]) will enrich these models with normative decision making.

We will continue to develop our simulator league team in the forthcoming years and we will implement the main concepts mentioned in this paper. We will also investigate whether it is beneficial to use several groups and leaders within a team and to which extent the agents should be autonomous related to the group.

References

1. H. Åberg, Å. Åhman, J. Andreasen, M. Boman, M. Danielson, C.-G. Jansson, J. Kummeneje, H. Verhagen, and J. Walter. "Ubu: Utility-based uncertainty handling in synthetic soccer" In: **Proceedings RoboCup98**, 1998.

2. D. Andre, E. Corten, K. Dorer, P. Gugenberger, M. Joldos, J. Kummeneje, P. A. Navratil, I. Noda, P. Riley, P. Stone, T. Takahashi, and T. Yeap. Soccerserver manual ver. 4 rev. 01. *http://www.dsv.su.se/~johank*, 1998.

3. M. Boman."Norms In: artificial decision making", **AI and Law**, 1999.

4. M. Boman, J. Andreasen, M. Danielson, C.-G. Jansson, J. Kummeneje, J. Sikström, H. Verhagen, and H. Younes. "Ubu: Pronouncers In: robocup teams" In: **Proceedings of RoboCup Workshop - PRICAI 98**, 1998.

5. M. Boman and H. Verhagen. "Social intelligence as norm adaption" In: B. Edmonds and K. Dautenhan, editors, **SAB Workshop on Socially Situated Intelligence**, Centre for Policy Modeling. Technical Report., 1998.

6. K.M. Carley and A. Newell. "The nature of the social agent", **Journal of Mathematical Sociology**, 19:221--262, 1994.

7. C. Castelfranchi. "Multi-agent reasoning with belief contexts: the approach and a case study" In: M. J. Woolridge and N.R. Jennings, editors, **Intelligent Agents**, Springer-Verlag, 1995.

8. R. Conte and C. Castelfranchi. **Cognitive and social action**, UCL Press London, 1995.

9. R. Conte, C. Castelfranchi, and F. Dignum. "Autonomous norm-acceptance" In: **Proceedings of ATAL 98**, 1998.

10. D. Dennett. **Brainstorms**, Harvester Press, 1981.

11. G. Dorais, R.P. Bonasso, D. Kortenkamp, P. Pell, and D. Schreckenghost. "Adjustable autonomy for human-centered autonomous systems on mars", presented at Mars Society Conference, 1998.

12. J.M. Epstein and R. Axtell. **Growing Artificial Societies - Social Science from the Bottom Up**, MIT Press, 1996.

13. M. Gilbert. **On Social Facts**, Routledge, 1989.

14. M. Gilbert. **Living together: rationality, sociality and obligation**, Rowman and Littlefield, 1996.

15. J. Habermas. **The Theory of Communicative Action. Volume One: Reason and the Rationalization of Society**, Beacon Press, Boston, 1984. transl McCarthy, originaly published as Theorie des Kommunikativen Handels, Suhrkamp, 1981.

16. N.R. Jennings and J.R. Campos. "Towards a social level charaterisation of socially responsible agents" In: **IEEE Proceedings on Software Engineering**, volume 144, pages 11--25, 1997.

17. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. "Robocup: The robot world cup initiative" In: **Proceedings of IJCAI-95 Workshop on Entertainment and AI/Alife**, Montreal, 1995.

18. Y. Moses and M. Tennenholtz, "Artificial Social Systems", Computers and AI, 1995.

19. P. Stone. **Layered Learning in Multi-Agent Systems**, Ph.D. thesis CMU-CS-98-187, Carnegie Mellon University, Pittsburgh, USA, 1998.
20. P. Stone and M. Veloso. "Task decomposition and dynamic role assignment for real-time strategic teamwork" In: **Proceedings of ATAL 98**, 1998.
21. M. Tambe, J. Adibi, Y. Alonaizon, A. Erdem, G. Kaminka, S. Marsella, I. Muslea, and M. Tallis. "Isis: Using an explicit model of teamwork" In: **RoboCup'97: Proceedings of the first robot world cup competition and conferences**, Springer Verlag, 1998.
22. R. Tuomela. **The Importance of Us: A Philosophical Study of Basic Social Norms**, Stanford University Press, 1995.
23. H.J.E. Verhagen and R.A. Smit. "Multiagent systems as simulation tools for social theory testing", presented at ICCS&SS Siena, 1997.
24. E. Werner. "Logical foundations of distributed artificial intelligence" In: G.M.P. O'Hare and N.R. Jennings, editors, **Foundations of distributed artificial intelligence**, Wiley, 1996.
25. H. Younes. Current tools for assisting real-time decision making agents. Master's thesis no. 98-x-073, DSV, Royal Institute of Technology, Stockholm, Sweden, 1998.

A COMPUTATIONAL MODEL FOR DECLARATIVE RECOGNITION AND IMPERATIVE ACTION

Thomas Weiser

Department of Computer Science
Technische Universität München
Arcisstr. 21, D-80333 München, Germany
email: weiser@in.tum.de

Abstract

Sita is a novel agent language for modeling intelligent behaviour in dynamic environments. The central modeling principle is the distinction of recognition and action as the two complementary basic skills essential for situated behaviour. The underlying computational model combines an incremental bottom-up logic programming mechanism with a concurrent process calculus. This principle allows the clear separation and natural description of declarative knowledge about situations and procedural knowledge about acting.

Keywords: *agent architecture, computational model, programming language, situated action, reactive logic programming*

1. Introduction

In this paper we propose a new computational model for intelligent agents.

Much work has been done on the theoretical basis of agenthood. Probably the best known is the BDI (Belief, Desire, Intention) approach [11], which builds on a temporal modal logic. The conceptual level of these theories is too abstract to directly derive concrete agents. Moreover, complete proof procedures are too complex to have a feasible implementation.

On the other hand, there is work on agent architectures, like InteRRaP [9], where functional models are used to describe the agent's behaviour, partitioned in several modules. The employed concepts are crude and still very abstract. Therefore, these architectures are more helpful for the abstract agent specification, but less helpful for the implementation.

According to these observations there are needs for concrete computational models supporting the agent design process from the implementation language side. Such models should lift the expressiveness of the implementation language to a level and a direction more suitable for agent design, compared to the presently most used languages like C++, Java or Prolog. These languages suffer from either being very low-level or having no support for reactivity. Narrowing the gap between the architectural level and the language level will substantially enrich the methodology of agent design.

In this paper we present such a novel computational model, which we call Sita (SITuated Action). The aims behind the development of Sita can be summarized as follows:

Support for reactive behaviour: An agent lives in a dynamic environment, which is unpredictable in principle. Therefore the agent must be able to trigger adequate, possibly nested reactions.

Reasoning on complex situations: The agents must classify their situations to know how to respond to them. This can be a complex reasoning task, which again needs to be performed in a reactive manner.

Coordination of concurrent activities: The agent's behaviour is composed of a varying set of execution threads. As these concurrent threads highly depend on each other, there must be mechanisms to coordinate these processes.

Precise semantics: A firm semantic basis simplifies the understanding. Furthermore it is a prerequisite for formal program verification (which could be a future development).

Feasible implementation: The computational model needs to have an effective and efficient implementation.

2. Recognize and Act

The central modeling principle of Sita is the distinction of *recognition* and *action* as the two basic skills essential for situated behaviour: Passively

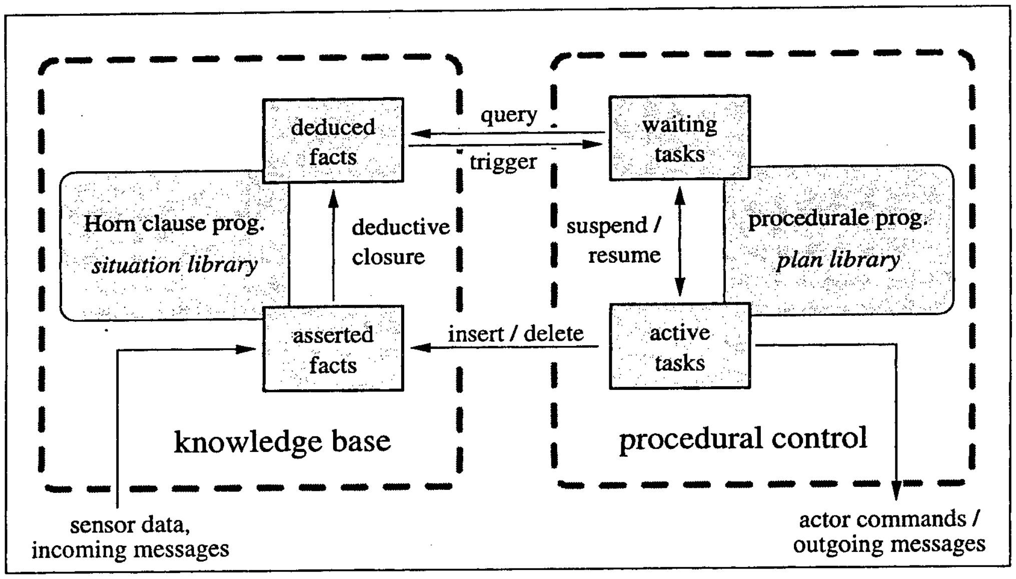


Fig. 1. Sita architecture

waiting for something to happen, and actively making something happen. Accordingly an agent is specified through describing the situations the agent should recognize at first, and the reactions the agent should respond with at second.

This distinction is rooted in the tradition of production systems like OPS5 and CLIPS, where programs are given by a set of condition-action-pairs [8]. An example for the use of a production system is the multi agent test-bed Magsy [3]. Each agent is an OPS5 interpreter extended by the capability of asynchronous message passing. Magsy has been used for building a distributed planner for flexible manufacturing plants [4] and for controlling forklifts in an automated loading-dock [10].

However, production systems suffer from two substantial drawbacks, which restrict their usefulness for the mentioned applications:

1. The rule selection process utilizes a very simple pattern matching concept with little expressive power. The condition parts of the rules are composed solely of fact patterns as primitives. There is no concept to abstract condition expressions under a new name. So one cannot compose complex expressions out of other expressions. Furthermore, this makes it impossible to use recursive formulae.
2. The action parts of the rules are simple sequences of actions without any control structures. Complex procedural operations have to be scattered to several rules, whereby the user is forced to manage the execution context on his own.

While preserving the advantages of production systems (reactive, symbolic, event-driven computation), Sita introduces new concepts both for the recognition and the action phase to overcome these drawbacks.

Two different formalisms are used for modeling the two basic skills (see fig. 1). On the one side there is a *deductive knowledge base*, responsible for monitoring the (internal and external) state. This is a process of analyzing the current situation, building an abstract world model, activating of corresponding goals and calling for adequate actions. To serve these complex tasks the Sita knowledge base employs a logic programming system based on Horn clause logic.

On the other side there is a *concurrent procedural language*, responsible for the description and coordination of the action threads the agent is in. It aims at a general but high-level model of coordinated computing. In contrast to production systems there is a powerful set of imperative concepts: sequential and parallel composition, guarded choice based on knowledge base queries, and definition of task abstractions by means of procedures.

These two components are linked through a common interface. Firstly, there are primitive actions to modify the knowledge base, i.e. the insertion and deletion of facts. Secondly, there are queries that trigger procedural actions upon entailment of the query expression.

According to this architecture, the agent's state is split into two distinct components: the facts present in the knowledge base which can be queried and updated by processes; and the task, representing the processes to be executed by the agent.

To be linked with the outside world the agent needs capabilities to perceive and to act. Perceived information is stored as messages in the knowledge base. External actions are effected through special primitives in the procedural part.

3. Knowledge Base

The Sita knowledge base uses Horn clause logic with negation as failure and function symbols in order to handle knowledge representation and abstraction, situation recognition, and decision making. It consists of a logic program, a fact base, and a forward-chaining inference machine.

The basic expressions of the logic language are predicates, which come in three flavors: Extensional predicates are containers for those facts that may be asserted or retracted through actions or perception. Intensional (or derived) predicates are defined by the clauses of the logic program and are interpreted by the deduced facts. Built-in predicates provide for some basic functions, e.g. arithmetic operations. Accordingly the fact base contains two sets of facts, asserted and deduced ones.

The inference engine continuously maintains the set of deduced facts in dependence of the current set of asserted facts and in correspondence to the logic program. This maintenance is an incremental and active reasoning process. All changes in the extensional part of the fact base will cause corresponding changes in the intensional predicates.

In the following example, it is assumed that `edge` is an extensional predicate. The two clauses define the transitive closure `path` based on the `edge` relation. Whenever the base relation changes, the derived relation will change accordingly. In this example, the `edge` relation may grow (or shrink) step by step through certain perceptions, while the `path` relation may continuously guide the agent's action to find a path between certain nodes.

```
path(X Y) ← edge(X Y).
path(X Y) ← path(X Z) path(Z Y).
```

The knowledge base is a pure *declarative* formalism, which gets its meaning through the well-founded semantics [14]. The advantages are e.g. that the ordering of the clauses within a program is irrelevant. The same goes for the ordering of the condition elements within the body of a clause. This is a big improvement compared to conventional logic programming systems like Prolog.

The evaluation is done by an incremental bottom-up algorithm, which is an essential property for obtaining reactive, event-driven situation recognition. To accomplish this, we extended the well-known Rete-algorithm [6] in several ways:

1. In the presence of recursively defined relations care must be taken about facts, that either support themselves or negate themselves. In the example above, if the relation `edge` shrinks, the relation `path` must shrink coherently, such that `path` is the minimal relation fulfilling the clauses. Therefore we have developed an appropriate reason maintenance algorithm, which is based on the idea of labeling each fact with the length of its shortest derivation.
2. We apply the magic set transformation [1], known from deductive databases. With this technique, instead of generating the whole set of consequences, only those parts of the program's model are evaluated that contribute to answering the current queries. In the example above, the relation `path` may be adorned `(in out)`, i.e. the first argument is marked as an input argument, the second as an output argument. This limits the computation of paths to those starting nodes that are needed in other parts of the program.
3. We found that the efficiency of the algorithm is highly dependent on the order in which changes are propagated through the clauses. Hence an appropriate scheduling mechanism is essential for efficient execution.

As these algorithmic aspects go beyond the scope of this paper, we omit the details here.

4. Procedural Language

The agent's knowledge base is complemented by a concurrent procedural language.

The current state w.r.t. procedure execution is represented by the agent's present task. A task is either an elementary action, a compound task, or the name of a procedure (see fig. 2).

The elementary actions are the insertion and deletion of facts, and sending a message to another agent. In addition certain builtins provide special actions, like commanding an effector, or accessing the operating system.

Tasks can be combined by sequential composition, parallel composition, and by guarded choice. The latter consists of a set of alternative task continuations, each guarded by a query expression. The task execution suspends until at least one of the expressions is entailed by the knowledge base. Thereupon the execution is continued with the body of the first entailed branch. The query expressions used as guards have the same syntax as clause bodies.

A procedure definition takes a task expression and makes it available under a given name. Procedures

elementary tasks:	**insert** *fact*
	delete *fact*
	send *address msg*
	$builtin\_action(args)$
procedure application:	$proc\_name(args)$
sequential composition:	$task_1\ task_2$
parallel composition:	$task_1 \ \|\| \ task_2$
guarded choice: .	$query_1 \Rightarrow task_1$
	$[]\ query_2 \Rightarrow task_2$
	$\vdots$
... with synchr. update:	$[]\ query_i \diamond mod_i \Rightarrow task_i$
priority assignment:	prio n
procedure definition:	proc $proc\_name(args)$:
	$task$
	end

Fig. 2. task expressions

may have formal arguments. They may also use local variables, which get bound by a query expression or a builtin action and can be used within the procedure. Procedures are not only used to structure the program, they are essential to express recursive task structures.

Through the use of the parallel composition multiple threads of execution are introduced, which allow for the required concurrency. A special statement is available to assign relative priorities to threads. This is useful to distinguish between, for example, an urgent reactive behaviour and a background planning task.

The presence of concurrency additionally requires a synchronisation construct, which is available in the context of guarded choice: branches of a choice may be written $query_i \diamond mod_i \Rightarrow task_i$, where mod_i is a sequence of elementary insert/delete-tasks. If the branch is chosen (on entailment of $query_i$), the sequence mod_i is executed immediately, without being interrupted by another thread. Semaphores and similar constructs can be implemented this way.

The mentioned procedural constructs should be viewed as the minimal required set. They confine themselves to the coordination aspects and leave the further computation to the knowledge base, i.e. the bottom-up reasoning process. As this may at times lead to somewhat clumsy programs, an implementation of Sita may add further procedural concepts, like assignment to local variables, looping constructs (beside recursion), or non-blocking knowledge base queries. But this doesn't affect the fundamental architecture of Sita.

5. Discussion

The Sita architecture combines a deductive knowledge base with a concurrent procedural control component. This structure reflects a basic model for intelligent agents. On the one side an agent has to maintain its current beliefs about the world and itself. This knowledge has to be represented on different abstraction levels. Higher levels model the agent's view of its situation and current goals. The Sita knowledge base is a tool to describe such abstraction processes by means of Horn clause logic in a purely declarative manner. This enables the agent to make reasoned choices. On the other side an agent has to change the world as well as its own beliefs and intentions. Procedures are a natural way to describe these active aspects. The presented combination of declarative and procedural concepts results in a novel programming model for reactive, intelligent agents.

We presented the Sita constructs on a programming language level. On top of that, the Sita architecture can also be viewed as a functional reactive agent architecture.

This gets evident if we switch to the usage of mental concepts.[1] Then the facts of the knowledge base represents the belief-state of the agent, the logic program is the agent's situation library, the procedures are its plan library. With the execution of a guarded choice the agent commits itself to a certain continuation task. Therefore the agent's current task can be seen as a representation of its intentions.

This correspondence between computational model and functional architecture indicates a good balance between generality and design support, and therefore contributes to bridge the gap between programming languages and agent design.

We have implemented most parts of the presented architecture. In particular we have a fast and complete implementation of the knowledge base, together with a subset of the procedural language. This has given us the ability to gather some first experiences regarding our approach of declarative bottom-up logic programming. First example applications are the n-queens problem, heuristic reactive search algorithms, and a blocks world planning system.

We have found that most aspects of these problems can be modeled declaratively (and nevertheless efficient), i.e. solely through the knowledge base. This confirms our assumption that the expressive-

[1] Compare with Shoham [12]: He defines an agent as an entity whose state is viewed as consisting of mental components such as belief, capabilities, choices, and commitments.

ness of the knowledge base formalism is sufficient to perform complex situation recognition tasks.

Related work: Many control architectures for the implementation of intelligent agents have been proposed [15]. We shortly compare the two well known systems Concurrent METATEM [5] and dMARS [2] with the Sita approach.

Concurrent METATEM is designed as an executable specification language. An agent is programmed by a set of rules of the form *past* $\Rightarrow$ *future*, using a propositional temporal logic. There is a forward-chaining reasoning process similar to that of Sita. The main difference is the representation of intentions: Concurrent METATEM doesn't apply explicit control structures like Sita's procedures; instead it maintains a set of future time formulae, for which the inference machine must find an execution path to make them eventually true. Concurrent METATEM and Sita have the principle of *declarative past and imperative future*[2] in common, which is also reflected in the programming paradigms of Sita's knowledge base and procedures.

dMARS is a control architecture for practical reasoning agents, which has its conceptual roots in the BDI theory [11]. Regardless of this different background the practical execution model has much in common with the one of Sita. Like Sita it uses ground facts for belief representation and a static plan library to keep procedural knowledge. Plans are triggered through belief changes, and plan primitives effect these belief changes. But there are also differences. Intentions are represented as explicit data structures with richer semantics compared to Sita tasks. Unlike Sita, dMARS supports backtracking on failure of plan execution. Situation recognition is handled in a simpler way, in that it doesn't support the expressive power of logic programming as Sita does.

The notion of *situated actions* was originally used by Suchman [13] in the context of human-computer interaction. She argues that (human) actions are never planned in a strong sense. Rather, actions are to a great extent linked to the specific situation at hand. Her observations underline the significance of an expressive situation description language.

Future work will focus on applications in more complex scenarios, where the need for intelligent reactivity is more evident. As a first step into this direction we are currently working on the integration of the Sita programming system into our robotics laboratory.

[2]introduced by [7]

Another research topic is the modeling of more complex agent architectures like InteRRaP [9], that include e.g. planning modules to gain deliberative behaviour. We expect to find effective mappings from these more structured but very abstract architectures onto the more concrete Sita concepts.

Consider a standard planning module for example. It may be implemented in Sita by a set of knowledge base clauses for a decision function to guide the planing process to the most promising direction, and a procedural loop that commits to that direction step by step. The planner may even adapt its search dynamically to changes in the current world model. More general, we suppose that situation recognition and coordinated action are two powerful building blocks for composing complex agent designs.

Bibliography

1. C. Beeri and R. Ramakrishnan. On the power of magic. In ACM, editor, *PODS '87. Proceedings of the Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 23–25, 1987, San Diego, California*, pages 269–284, New York, NY 10036, USA, Mar. 1987. ACM Press.
2. M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dMARS. In M. P. Singh, A. Rao, and M. J. Wooldridge, editors, *Proceedings of the 4th International Workshop on Agent Theories, Architectures, and Languages (ATAL-97)*, volume 1365 of *LNAI*, pages 155–176, Berlin, July 24–26 1998. Springer.
3. K. Fischer. The rule-based multi-agent system MAGSY. In *Proceedings of the CKBS'92 Workshop*. DAKE Centre, University of Keele, UK, 1993.
4. K. Fischer. *Verteiltes und kooperatives Planen in einer flexiblen Fertigungsumgebung*, volume 26 of *DISKI, Dissertationen zur Künstlichen Intelligenz*. Infix, St. Augustin, Germany, 1993.
5. M. Fisher. A survey of concurrent METATEM: The language and its applications. In D. M. Gabbay and H. J. Ohlbach, editors, *Proceedings of the 1st International Conference on Temporal Logic*, volume 827 of *LNAI*, pages 480–505, Berlin, July 1994. Springer.
6. C. L. Forgy. Rete: A fast Algorithm for the Many Patterns/Many Objects Pattern Match Problem. *Artificial Intelligence*, 19(1):17–37, Sept. 1982.
7. D. M. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proceedings of the Conference on Temporal Logic in Specification*, volume 398 of *LNCS*, pages 409–448, Berlin, Apr. 1989. Springer.
8. T. Ishida. Parallel, distributed and multi–agent production systems – A research foundation for distributed artificial intelligence. In V. Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*, pages 416–422, San Francisco, CA, 1995. MIT Press.
9. J. P. Müller. *The design of intelligent agents: a layered approach*, volume 1177 of *LNAI*. Springer, New York, 1996.
10. J. P. Müller and M. Pischel. The agent architecture inteRRaP: Concept and application. Research Report RR-

93-26, Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, Germany, 1993.

11. A. S. Rao and M. P. Georgeff. Modeling Agents Within a BDI-Architecture. In R. Fikes and E. Sandewall, editors, *Proc. of the 2rd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, Cambridge, Mass., Apr. 1991. Morgan Kaufmann.
12. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, Mar. 1993.
13. L. A. Suchman. *Plans and Situated Actions: The Problem of Human-Computer Communication*. Cambridge University Press, New York, 1987.
14. A. van Gelder, K. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, July 1991.
15. M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

An approach to the Development of a Knowledge Representation and Processing System with the Use of Agent-Based Technique[1]

Yuriy A. Zagorulko[1], Ivan G. Popov[2], Olga B. Karakozova[3],

[1]*Institute of Informatics Systems SB RAS,*
Lavrentiev str. 6, Novosibirsk, 630090, Russia
e-mail: zagor@iis.nsk.su
[2]*Russian Research Institute of Artificial Intelligence,*
Lavrentiev str. 6, Novosibirsk, 630090, Russia
e-mail: popov@iis.nsk.su
[3]*Russian Research Institute of Artificial Intelligence,*
Lavrentiev str. 6, Novosibirsk, 630090, Russia
e-mail: ok@iis.nsk.su

Abstract

An approach to the development of knowledge representation and processing system based on integration of classical knowledge representation means with methods of constraint programming and agent-based technique is considered. In contrast to other constraint programming systems, this one allows us to operate with imprecisely defined (subdefinite) values. To increase efficiency of the processes of logical inference and data processing, agent-based technique is used instead of a traditional production rule technique. Due to a natural combination of data-driven and event-driven mechanisms, the development of efficient intelligent systems for various applications is provided.

Keywords: *Object, agent, constraint programming, subdefinite computational model, data-driven and event-driven mechanisms, knowledge-based system.*

Introduction

Nowadays, the main trend in artificial intelligence changes from the opposition of different means and methods for knowledge representation and processing to their rational combination within a single system. This trend is based on understanding of the fact that each of these means executes its function and has its application domain. On the other hand, none of the well-known means for knowledge representation and processing can meet all needs of creating a real-life application system on its own. Let us consider the main of these means.

One of the most popular knowledge representation means is *semantic network*. Due to such properties as high associativity and flexibility for representation of information, semantic networks are considered to be a universal storage for any information (knowledge or data), that can be represented in terms of objects and relations between them. These properties have made semantic networks very popular. However, high flexibility of semantic networks is provided by means of a rather low level of knowledge representation that leads to complicated description of a problem, and thereby makes the work of the knowledge engineer (the user) extremely difficult.

The use of *frames* and *object-oriented approach* allows one to raise greatly the level of the knowledge representation language and the possibility of its customization in a concrete subject domain. The main advantage of such an approach is the possibility of linking locally every frame with its properties. This saves one from the necessity to take care of small details at the global level. Frame-based representation languages have the mechanism of demons and attached procedures which allows one to define functional dependencies for values of the slots. But, as a rule, in frame languages such dependencies are given at a low level, for example, in terms of procedures and functions, which makes the knowledge engineer to turn to skilled programmers for help. It would be better to use for this purpose the

[1] This work is supported by Russian Foundation for Basic Research (grant ¹ 99-01-00495)

mathematical and logical formulae or other means of high level.

The formalism of *production rules* is regarded as a powerful tool for expressing the operational semantics of the notions of the subject domain and for logical inference. It is characterized by a natural specification of knowledge, simplicity of modification and extension, and natural modularity. But production systems are oriented mostly to symbolic computations, therefore they are very difficult to apply to the solution to problems requiring numerical calculations. Another shortcoming of the production systems is that this technique usually use an expensive process of pattern matching that leads to overall inefficiency of applications created on this basis.

Another mean for organization of logical inference and data processing is the *agent-based technique* [1,2,3] which has become very popular among researchers in various fields of computer science. In contrast to production systems, agents act more autonomously and provide us with possibility to specify inference processes locally. Therefore application of this approach for knowledge representation and processing appears to be very attractive and fruitful.

In the framework of the paradigm of *constraint programming* [4], which is also very popular now, it is possible to specify problem in the form of a set of *constraints* over the values of parameters of objects (or problem). This approach is particularly convenient when the constraints can be represented by ordinary Boolean expressions. However, this approach allows one to solve a rather narrow class of problems which are reducible to the constraint satisfaction problem. Consequently, in practice constraints are often used in combination with other, more universal means of knowledge representation and processing. Application of this approach in AI languages appears to have a future. The methods of constraint programming, while usually used in numerical computations, complement nicely the traditional AI tools oriented at symbolic processing and declarative knowledge representation.

It should be noted that none of the above techniques has provided the means for working with imprecisely defined values and objects. Such facilities can be found in the method of *subdefinite computational models,* proposed by A.S. Narin'yani [5,6].

In this paper we consider an approach to the development of a programming environment for intelligent system design which is based on an integrated model of knowledge representation that unifies the classic means with the methods of constraint programming and agent-based technique on the basis of object-oriented technology.

In contrast to other systems also using constraints programming technique, this system allows us to operate objects with imprecisely defined values of slots. Refinement of such values is carried out by means of constraints linked to objects. The constraint satisfaction process is implemented using modification of the method of subdefinite computational models [7]. The key feature of this modification is linking constraints with objects, which enables one to work with a dynamic set of constraints.

Another important feature of the system is that it utilizes the agent-based technique instead of the production system traditionally used in inference and data processing control. Unlike production systems, the agent-based technique makes the inference and data processing more efficient by using the event-driven mechanism. According to this idea, activation of each agent is performed by an associative data-driven process where an agent reacts only to an event related to it (e.g. appearance of new objects or setting new relations between them or changing values of the object's attributes).

Due to the natural integration of both data-driven (constraint-based technique) and event-driven (agent-based technique) mechanisms, the presented system can be used to create highly efficient intelligent applications, in particular, applications concerning the natural language processing or requiring the combination of logical inference and computations over imprecise values.

First, we consider the method of subdefinite computational models as it is of particular importance for our approach.

2. The method of subdefinite computational models

Let T be an "ordinary" data type with the set of values A and the corresponding set of operations over A. We denote by A^* the set of all subsets of A. Elements of A^* will be called *subdefinite values* or *SD-values* and denoted by a^*. The values a^* containing only one element of A will be called *exact values*. A special value equal to the entire set A will be said to be *fully indefinite*, and a value equal to the empty set will be called *inconsistent*.

For each operation $P: A^n \rightarrow A$ of type T we can define the correspondent operation $P^*: A^{*n} \rightarrow A^*$ as a subdefinite extension of the operation P:

$$P^*(a_1^*, ..., a_n^*) = \{P(a_1, ..., a_n) \mid a_1 \in a_1^*, ..., a_n \in a_n^*\}.$$

These new operations have similar semantics but may be applied to subdefinite values and the result of each of them is, in general, a subdefinite value too. So we can build a *subdefinite data type* T^* on the base of the original "exact" data type T.

Presently, subdefinite extensions have been constructed for various data types: *integer*, *real*, *symbolic*, *logical*, *sets*, etc.

Subdefinite data types can be used to represent uncertain data in a problem which is specified in terms of *constraints* on its parameters. The method of subdefinite computational models is used to solve such problems.

First, we consider as constraints are represented in subdefinite computational models.

Formally, *a constraint* is a Boolean expression $C(v_1,...,v_n)$ that is required to be true. The variables $v_1,...,v_n$ linked by the constraint may be of any subdefinite data types.

Each constraint must have *functional interpretations*. This means that the constraint can be represented by a set of functions:

$$f_i^* : A^{*(n-1)} \to A^*,$$

which are called *interpretation functions*. Each of these functions allows us to calculate the value of one variable from the values of the other ones:

$$v_i = f_i^*(v_1,...,v_{i-1}, v_{i+1},...,v_n).$$

For example, the constraint

$$U = I * R \qquad (1)$$

can be interpreted by the following three interpretation functions:

$$U = f_1^*(I, R),$$
$$I = f_2^*(U, R),$$
$$R = f_3^*(U, I),$$

where

$$f_1^*(x, y) = x * y,$$
$$f_2^*(x, y) = x / y,$$
$$f_3^*(x, y) = x / y.$$

Here '*' and '/' denote subdefinite extensions of arithmetical operations of multiplication and division, respectively.

If an expression in a constraint is too complicated, it is possible to simplify it by adding new variables and splitting the complex constraint into several simpler ones. For example, the constraint

$$Y = (X + C)^2 \qquad (2)$$

can be divided into two more simple constraints:

$$S = X + C, \qquad (3)$$
$$Y = S^2, \qquad (4)$$

where S is an auxiliary variable.

A *subdefinite computational model* (*SD-model*) is represented by a bipartite oriented graph (*subdefinite functional network*) and a discipline of its processing, or *data-driven computations*. There are two types of vertices in a subdefinite functional network: variables and interpretation functions. The incoming edges of a function vertex connect it to the variables whose values are input arguments of the function. Outgoing edges of the function vertex point to variables in which store the results produced by the function.

For example, the subdefinite functional network corresponding to the constraint *(1)* is illustrated in Figure 1.

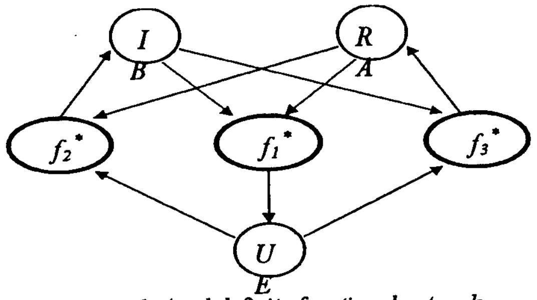


Figure 1. A subdefinite functional network

The principle of data-driven computations means that a change of the value of variable vertices activates (causes execution of) the function vertices; execution of the function vertices, in turn, may cause a change of the resulting value of variable vertices, and so on. If at least one variable gets an inconsistent value (empty set), the process will be stopped and the *SD-model* will be considered inconsistent.

During the process of interpretation a new value of the variable is calculated and intersected with the old one. Since it is the result of this intersection that is assigned to the variable, its value can be only refined, i.e. the new value is a subset of the old one. It should be noted that the value of the variable is considered to be changed only if it is actually refined.

As it was shown in [7], for all data types containing only a finite set of *SD-values*, this algorithm terminates in a finite number of steps. In the case of infinite sets of *SD-values* (for instance, intervals of real values), the stopping criterion can be based on the preset threshold of computation accuracy ε. This threshold ε determines the maximum possible distance between two values for which they are still considered to be identical.

3. The base means for knowledge representation and processing

The base tool used to represent declarative knowledge in our system is *a semantic network*, which consists of objects linked by binary relations. Besides, it is possible to specify knowledge in the form of a set of *constraints* over the values of parameters of objects.

3.1. Objects and relations of the semantic network

An *object* can be any entity of the subject domain, defined by the knowledge engineer. Objects with the same properties are combined into one *class*. The properties of the class define the names and types of values of slots (attributes) of objects, possibly their default values, and their behavior. The latter is determined by the set of constraints which are defined on the values of the objects' slots.

The values of slots can be characters, strings, atoms (indivisible strings), integer and real numbers, tuples and sets. An object may be the value of a slot too. An important feature of objects is that their slots may be subdefinite, i.e., their values may be subsets of the domain of admissible values. Subdefinite values can be defined as intervals of values for numerical data types and as sets of possible values for other types.

When an object is created, the slots whose values are not given will be filled with subdefinite values. If type of the slot values is elementary (characters, strings, atoms, or numbers), then the slot will be given a subdefinite value which is the entire set (domain) of admissible values. If type of the slot values is a tuple, set, or object, then the slot will be filled with the completely indefinite value *'?'*.

Classes may inherit properties of other classes (in this case, the former are called subclasses, and the latter are superclasses), with a possibility of multiple inheritance. Some properties of superclasses can be redefined in subclasses.

If a subclass inherits from several superclasses containing slots with the same name, then these slots are "glued" into a single slot. The types T_i of these slots specified in the superclasses must be the same or at least they must be derived from the same base type T_0 and have a non-empty common subset of values, i.e., $\cap\, dom(T_i) \neq \varnothing$. In this case the slot in the subclass will have type T derived from T_0 and $dom(T) = \cap\, dom(T_i)$.

In the definition of a subclass we can redefine the type of an inherited slot. The new type T of the slot must refine the old type T_0, i.e., $dom(T) \subset dom(T_0)$.

Any constraint in an objects class definition can be labeled. Use of labels of constraints makes it possible to redefine this constraint through the process of inheritance.

A *binary relation* is treated as a special object with two slots, which are called *relation arguments*. We can define constraints for relations just as we do it for objects. Note that in this case constraints are defined on the values of slots of objects that are arguments of relation.

We have selected binary relations as a particular class of relations because they have useful properties like *reflexivity*, *symmetry*, *transitivity*, which can be built into the system.

3.2. Functional network

As said above the system allows one to bind with any object a set of constraints defined on the values of its slots. Since the value of a slot can be an object, constraints can link the values of slots from several objects.

The constraints associated with some object constitute its local *SD-model* and make it possible to automatically refine subdefinite values by means of the mechanism of constraint satisfaction. The method of subdefinite computational models is used to implement this mechanism.

The set of SD-models of all objects presented in a semantic network constitutes a functional network, which is activated for every modification of the objects; it ensures recalculation and modification of the values of slots of the related objects.

3.3. The base operations

The system includes operations for creating objects and instances of binary relations (**new**), editing them (**edit**) and deleting them from the network (**delete**). At the same time with the edition of the semantic network, the functional network is modified.

So, the **new** operation creates objects and instances of binary relations and inserts their into the semantic network, and simultaneously adds constraints bound with the object or relation to the functional network.

After objects or relations are deleted, the semantic network and the functional network are corrected. In the semantic network, all references to the deleted objects are replaced by the completely indefinite value *'?'*. At the same time, all constraints related to the deleted objects and relations are removed from the functional network.

Thus, the functional network can be modified during the application system operation as a consequence of both insertion of new objects and relations into the semantic network and edition of the existing ones. This takes place because the process of setting up new links among objects can lead to involving into the functional network new constraints which include slots of these objects.

It should be noted that the operations described above leads to the immediate activation and execution of the functional network until its stability is achieved.

4. The agents system

The process of inference and data processing are defined as *an system of agents* working over the semantic network.

An agent is the object of special type. Agents may inherit properties of other agents. Some properties of base agents can be redefined in derived agents.

Each agent like one in [3] responds only to the related events. The response of the agent is in an execution of certain predefined actions. Specifically, the agent can create new objects in the semantic network, or change values in the slots of the existing objects, or set up some new relation between them. From this point of view agents work like productions that define inference and data processing in the traditional knowledge based systems. However, unlike the production systems that use an expensive pattern-matching routine, the activation of agents is based on the event-driven mechanism that significantly increase an efficiency of the inference and control processes. Besides agent-based approach allows one to implement natural integration of both data-driven and event-driven techniques.

The structure of agent is similar to the structure of the usual object and includes the following components:

- *Name*
- *Name of base agent*
- *External slots*
- *Internal Slots*
- *Condition*
- *Operators*

Here, *Name* is the agent's own name which is the unique reference to the agent.

Name of base agent is a name of agent whose properties are inherited.

External slots define links of this agent with the outside world. The description of the agent specifies name of the link (i.e. slot name) and its type. The latter is a name of a class of the objects that are "visible" to the agent through the link. In each instance of the agent the values of its external slots are references to the appropriate objects in the semantic network.

Internal slots of agent determine its state. One part of the internal slots is accessible for other agents, whereas another part of them is used internally and is inaccessible from the outside.

Condition is represented by a set of constraints defined on the agent's slot values. The truth of this condition ensures actuation of the agent, i.e. execution of the actions predefined in *Operators*.

Actions defined as the *Operators* may be either mentioned above operations with objects in the semantic network (creation, edition or deletion) or certain system operations which are used to provide data input and output, organization of user interface and so on.

Like the usual objects, agents are created with the help of **new** operator. Operators **edit** and **delete** also may be applied to the agents.

The overall set of agents constitutes an associative structure over the semantic network, called *agent network* (Figure 2).

Each agent reacts to the appearance of new objects, that correspond to its external slots. If the new created object is "visible" to the agent and all of its external slots are associated with the other "visible" objects, then condition of agent actuation is to be tested. When the condition becomes true, the agent is executed and its operators are performed. An agent reacts also to other events related to objects corresponding to its external slots, e.g. changing values of the object's slots or setting new relations between the objects.

Since values of the objects' slots generally may be subdefinite, the truth of the agent condition may be subdefinite too. In this case the local computational model is formed from the agent's constraints specifying by the values of the slots of correspondent objects and is added to the functional network, and the execution of the agent is postponed up to the moment when the values of the objects' slots are refined to such a degree that condition is exactly true. The recalculation of these constraints will be performed under each interpretation of the functional network as it takes place for usual objects. When values of the objects' slots will be refined and the truth of the condition will be calculated exactly, this local computational model will be deleted from the functional network.

Actuation of the agent can lead to creating new objects in the semantic network or changing state of the existing ones. This, in turn, causes activation of the other agents associated with the new or modified objects and so on. According to this process, agents are activated concurrently and asynchronously, as the objects appear or change in the semantic network. Therefore the event-driven mechanism of agent activation is based on the

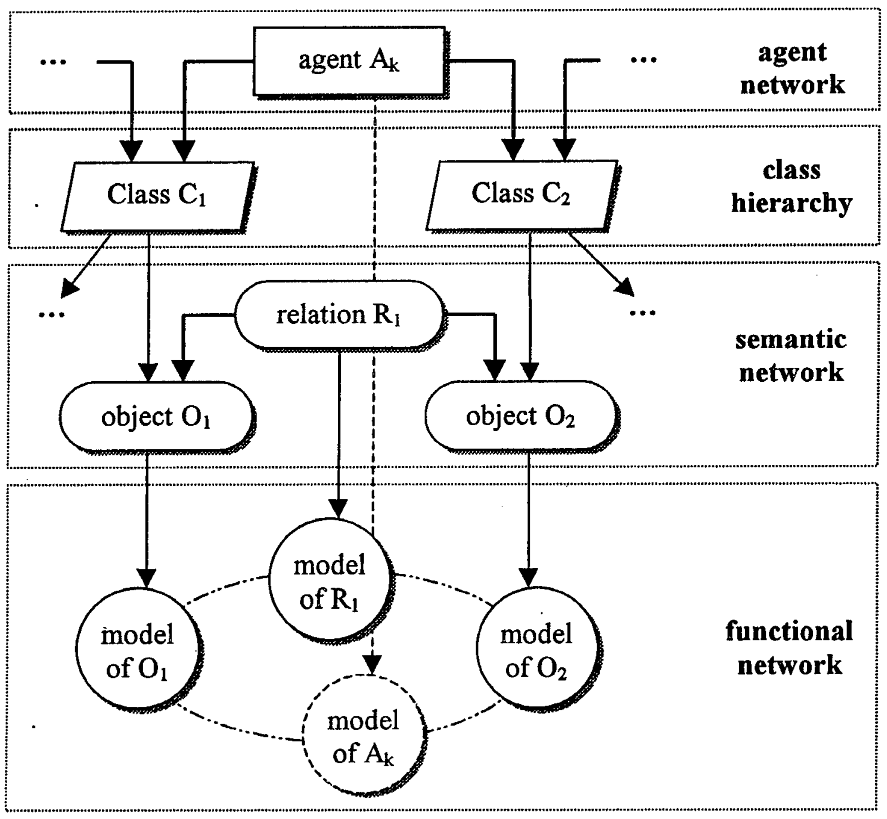


Figure 2. Four levels of knowledge representation

general data-driven mechanism, discussed in the chapter 2.

5. Control of the agents system

As the same object can be associated with several agents, the creation of this object may involve an activation of a set of agents. This set will be called traditionally as the conflict set. Since generally the final result of the inference process depends on the order of agents actuation, an agent priority mechanism is used to control agent execution from the conflict set. According to this mechanism each agent may be given by a certain number that determines its execution priority. Under the processing of the conflict set agents with the high priority are to be processed first. The use of the priorities technique allows us to avoid partially collisions and contradictions in the inference process.

Another way to solve the above problem is defining a hierarchy of agents. According to this approach agents which are located at the bottom of the hierarchy and intended for processing of special cases, get a highest priority, while agents that are placed at the head of hierarchy and oriented to processing of more general situations get least one. In this case specific agents will be activated first. As regards more general agents, they will be activated only if current situation does not allow specific agents to be actuated.

Activation of agents can be also managed by dividing all set of agents into separate groups and activating these groups selectively. Agents of the active group act in the usual way, i.e. they start working when suitable objects appear or some objects change their values. The agents of the inactive groups therewith only wait for appearance or modification of objects in the semantic network and just keep references to these objects but do not perform any actions. Actuation of such agents is postponed up to the moment when their group becomes active. After the inactive group is activated, all its agents function over the combinations of the objects kept before and afterwards they will be switched to the conventional waiting state.

The division of the overall set of agents into separate groups allows one to split the main task into the separate subtasks. Furthermore, reduction of the set of active agents increases the efficiency of the overall inference process. Since the activation and deactivation of the groups of agents can be performed by the agents themselves, the system gets control over the inference process. Thus, it can be regarded as self-organizing.

6. Architecture of the system

The presented program environment consists of specification facilities and a program kernel.

The specification facilities include the knowledge representation and processing language, as well as the interactive graphical interface which allows one to construct interactively classes of objects and relations, user-defined types, and specify agent systems.

The definition of classes and relations are stored in special libraries of notions. All these libraries have value of their own, can be used separately by other users in the development of other applied systems.

The agent systems is specified by the means of the knowledge representation and processing language, based on a developed or inherited library of notions.

The program kernel of the knowledge-based system (Figure 3) includes the agent control system, a semantic processor (semantic engine), and a data driven processor (subdefinite engine).

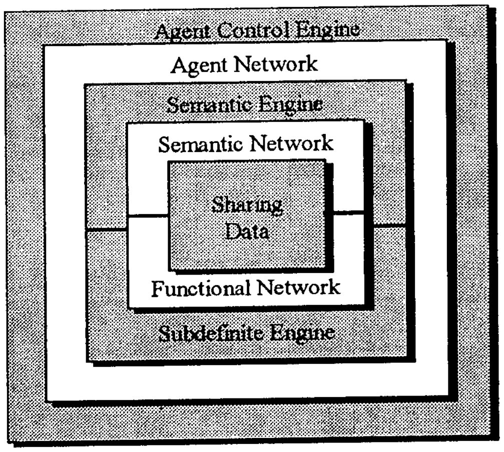


Figure 3. A scheme of the kernel of the knowledge-based system

The semantic engine performs operations over the semantic network (creation, modification, and deletion of objects and relations, pattern matching) and data of other types (numbers, strings, tuples, sets, etc.).

The subdefinite engine is responsible for interpretation of computational constraints on the values of the objects' slots.

The agent control system supports the entire process of execution of the agent system, from agent activation to actuation.

The (interrelated) semantic and functional networks play the role of the system's main storage. The semantic network represents the declarative knowledge on objects and relations between them, while the functional network represents computational relations (constraints) between the objects' slots. The form of the semantic and functional networks is determined by the notions that had been introduced by the knowledge engineer at the stage of designing classes of objects and relations.

The agent system is the part of the system that defines (together with the functional component) the procedure for logic inference and information processing.

7. Advantages of the use of agents

The use of agents instead of production rules for definition of logical inference and knowledge processing provides user (knowledge engineer or practical linguist) with a lot of serious advantages.

First, agent is an object, therefore one can operates agents as the usual objects. In particular, agents can be organized into hierarchy, using all advantages of inheritance of their features.

Let us consider such field as diagnostic expert systems. Usually, the diagnostics of an object is provided by set of production rules, each of which corresponds to the specific case of a disease or damage determination. If there is a lack of symptoms, the more general production rules are activated and make general diagnosis and recommendation. Thus, the set of all production rules of the diagnostic expert system is divided into various groups, according to the depth and generalization of their diagnostic possibilities.

Similarly, the set of agents can be organized into hierarchy with the "general" agents at the bottom and "specific" ones at the top. In this case the low-level agents are able not only to analyze additional information, but also to perform additional actions. While agents of the highest levels of hierarchy can be used as for performing diagnostics by the general way and so far initialization of processes of inquiring or refining missing symptoms, generating hypothesis and so on.

There is another advantage of this approach. Organization of hierarchy of agents allows one to solve a problem of the choice between agents from the conflict set and save the knowledge engineer from the laborious and often not so clear assignment of numeric priorities to agents. Thus the declarative information about the position of agent in the hierarchy defines by the natural way the operational characteristic of the agent – priority of its activation and actuation.

The use of agents is also very suitable for natural language processing. In this case each agent may serve for the recognition of the specific syntactic or semantic construction of the text. The input words are placed sequentially in the semantic network in the form of new lexical objects and activate the corresponding agents immediately.

Notice, that some characteristics of these words can be undefined or subdefinite and agents will not be actuated until these characteristics are enough defined or refined by the other agents. Under this approach agents intended for both syntactic and semantic analyses can work simultaneously to recognize language constructions.

For example, consider the agent that recognizes word collocations. If not all characteristics of the words potentially included in this collocation are defined, the agent waits until information about each word is refined to such degree, that they can be recognized as parts of the collocation which correspond to some notion of the subject domain.

As the agents act locally and respond immediately to the input of new words agent-based technique allows one to process phrases of natural language text actually in the real time and hypothesize before the phrase is received completely. This feature can be useful, for example, for the speech recognition systems or speech control of technical devices (robots).

Besides, utilization of agents instead of production systems gives us a lot of advantages. In particular, productions usually use an expensive process of pattern matching that leads to overall inefficiency of applications created on this basis. To accelerate the pattern matching routine, some associative methods such as RETE-algorithm [8] are applied. But this does not solve the problem because these methods work fine only for regular and static systems where the set of productions is stable. Whereas activation of agents is performed by an associative data-driven process where each agent reacts only to an event related to it, that allows one to increase efficiency of the processes of logical inference and data processing.

8. Conclusion

The presented system is the development of the program environment [7] based on an integrated model of knowledge representation. Due to unifying such means as frames, semantic networks, production rules, subdefinite computational models, and constraint programming techniques, it can be used to create a broad range of intelligent systems requiring the combination of logical inference and computations over imprecise values.

In contrast to its predecessor [7], this system uses the agent-based technique for inference and data processing control instead of a production system. High efficiency of these processes is achieved by using the associative event-driven mechanism instead of an expensive pattern matching routine. So, each agent reacts only to events related to it, e.g. appearance of new objects or setting new relations between them or changing values of the object's attributes.

This event-driven mechanism is primarily based on the data-driven mechanism used in [7] as the constraint-satisfaction technique for refining imprecisely defined values. Due to such a natural integration of data-driven and event-driven methods the agent-based, technique can be used to operate imprecisely defined values.

Aside from providing additional functional and descriptive capabilities, the presented program environment greatly increases productivity of a knowledge engineer designing an application and makes it possible to form the knowledge base in a rather natural, high-level manner. After defining the necessary system of notions, one can process objects in terms of agents, without worrying about their internal semantics. The knowledge engineer can concentrate mostly on their properties and interrelationships.

The presented system can be used to create highly efficient intelligent applications, in particular applications concerning natural language processing, creation of expert systems and problem-oriented shells, design of sophisticated knowledge-based systems, creation of planning and decision support systems, and intelligent systems for controlling complex objects, including robots.

Bibliography

1 M.Wooldridge and N. Jennings. Agent Theories, Architectures, and Languages: A Survey. Intelligent Agents - ECAI-94 Workshop on Agent Theories, Architectures, and Languages (LNAI Volume 890), , Amsterdam, 1994, pp. 1-22.

2 M.Wooldridge. Issues in Agent-Based Software Engineering. In: Cooperative Information Agents: First International Workshop, CIA-97 (LNAI Volume 1202), Springer-Verlag, Berlin, 1997, pp. 1-18.

3 C.Meghini. A reactive logical agent. Ibid., pp. 148-158.

4 Mayoh B. Constraint Programming and Artificial Intelligence, Constraint Programming. Springer-Verlag, 1993. NATO ASI Series F: Computer and Systems Sciences, Vol. 131, pp. 17-50.

5 Narin'yani A.S., Sub-definiteness and Basic Means of Knowledge Representation, Computers and Artificial Intelligence, 1983, Vol. 2, N 5, pp. 443-452.

6 Vitaly Telerman, Dmitry Ushakov., Data Types in Subdefinite Models. In: Jacques Calmet and others (eds.), Art. Intell. and Symbolic Mathematical Computation, Lecture Notes in Computer Science; Vol. 1138, Springer, (1996), pp. 305-319.

7 Yu.A.Zagorulko, I.G.Popov. Object-Oriented Language for Knowledge Representation Using Dynamic Set of Constraints. In: Knowledge-Based Software Engineering, P.Navrat, H.Ueno (eds). -(Proc. 3rd Joint Conf., Smolenice, Slovakia). -Amsterdam: IOSPess, 1998, - pp.124-131.

8 Forgy C., RETE: A fast algorithm for the many pattern/many object pattern match problem // Artifical Intelligence, 1982, Vol. 19. pp.17-38.

PART III
Position Statements

AN INFRASTRUCTURE FOR A GENERAL, HIGHLY DECENTRALIZED WORKFLOW USING MOBILE AGENTS

Zoran Budimac, Mirjana Ivanović, Aleksandar Popović

Institute of Mathematics, Faculty of Science, University of Novi Sad,
Trg D. Obradovića 4, 21000 Novi Sad, Yugoslavia
e-mail:{zjb,mira,ror}@unsim.ns.ac.yu

Abstract

The paper describes an infrastructure for implementation of workflow management system using mobile agents. The usage of mobile agents in modelling and implementation of a workflow is a novel approach and simplifies the workflow management. Besides, the suggested infrastructure introduces fresh ideas in the field of mobile agents as well.

Keywords: *Workflow, Mobile Agents, Distributed Programming.*

1. Introduction

Using the workflow management system is a modern trend in modelling and implementation of an information flow and business processes in a company [7,4]. Well organized and fully implemented, a workflow can replace a large part of a classical information system of a company. In a focus of a workflow is "work" (information, a task, a document, announcement, data, etc.) that flows through different points in a company. Deadlines and conditions of a work transition are defined in order to achieve the flow of works. When a condition is met, work is transferred to a next stage, where the next condition has to be fulfilled. At the end of its itinerary, work is done.

According to most general definition, a mobile agent is a program that is able: to stop executing at one node in a computer network; to transfer itself to another node in a network; and to continue execution there. More precise definitions include additional requirements for a piece of code to be a mobile agent. All definitions however, stress the important feature of mobile agents - autonomous behaviour. Mobile agents are a very fresh research area in the field of a distributed programming and artificial intelligence (see for example [6, 8, 9, 10].) Advantages of mobile agents with respect to classical techniques of distributed programming are numerous. Among many, we stress the following ones: potentially better efficiency of the whole system and a greater reliability.

In this paper, data structures, agents, tools, and principles for implementation of a workflow management system using mobile agents are presented. This approach has several main advantages over other approaches in organization and implementation of workflow management systems (including implementations using static agents):

- Uniformity of organization and implementation. Mobile agent system is used for implementation of: the workflow administration and monitoring tools, workflow client applications, and workflow enactment services [7]. Moreover, agents are used as a uniform interface to invoked applications and other workflow enactment services.
- Simpler flow management. Mobile agents are work-items that are passed to different users and autonomously take care of their current position and further itinerary.
- Flexibility. The flow of work is easily changed when using mobile agents. It is done only by changing agent itineraries or by introducing new agents. Organizational structure of a company is contained implicitly in agent itineraries and is easily changed.
- Scalability. The workflow implemented with mobile agents can easily grow with the underlying company or complexity of its business processes. In those cases just more agents are added, without the need to change and understand the rest of the system.
- Support of unstructured business processes. With the use of mobile agents, business processes must not be well structured.

The rest of the paper is organized as follows. In the next section, a basis for creation of individual work-agents is presented. A work-server that hosts work-agents is introduced in the third section. In order to ease the application of the workflow system and to increase its reliability, it is necessary to implement additional tools and specialized administrative agents. These are discussed in section four. In the fifth section, a reliability and protection of the proposed system are shortly discussed. In the sixth section, the related work is presented, while the seventh section concludes the paper.

2. Class *Task* and work-agents

Class (in the sense of object-oriented programming) *Task* represents an abstract (i.e., every) work in the proposed workflow system. This class is a descendant of a class that represents a mobile agent in a chosen mobile agent system. Objects of the class *Task* thus becomes mobile as well. The class contains the following attributes and methods: attribute containing work identification, attribute containing work deadlines, attribute containing an owner of work (the person that created the work), method for work externalization (to save the work into a file), method for work internalization (to load and recreate the saved work), itinerary, and a method for presenting a user-interface.

The itinerary is a list of triples of the following form: *(node, condition, methods)*. It represents a flow of work-agent through a network. A *node* represents an address of a node where the work will transfer itself from the current node. Only methods enlisted in the list *methods* are active on the current node. The work-agent will transfer itself to the *node* when and if the *condition* (a logical function) is fulfilled. If an itinerary of a work-agent can contain alternative routes through a network, then the itinerary must be represented as a tree, rather than as a list.

3. Work-server and work-host

Up to now, our system consists of work-agents that represent concrete tasks and are created by inheriting an abstract class *Task*. Besides work-agents, a work-server is needed as well. Work-server is a program that executes all the time on every node that is in the workflow system. Its basic tasks are the following:

- Listens the designated port, waits for incoming agents, internalize them, puts them into a list of agents on that host, and activates them.
- Inform the user about an arrival of new work.
- For every work-agent periodically calls the function *condition* for the current node.
- If the work-agent is too long on the same node, alerts the user.
- At the end (before switching off the computer), externalizes all agents that are currently under its supervision.
- At the beginning of its execution (after the computer is switched on), internalizes all saved agents.

The work-server is kept small and simple. All other necessary administrative functions are implemented as separate, specialized agents.

Besides work-server, every node contains one work-host that can be implemented as a stationary agent. The role of work-host is to offer the user basic data about all work-agents that currently reside on the node. The work-host is very simple - it just uses appropriate work-agent attributes and invokes its methods.

4. Other utilities and agents

Work-agents, work-servers, and work-hosts are the only software components that are really necessary for the proposed workflow system. However, if the system is to be more user-friendly, more reliable, and more flexible, additional tools and specialized agents are needed.

4.1. Templates and template library

The most important part in a definition of a new work-agent (at least from the workflow point of view) is a definition of an itinerary. For every class of a work-agent, a template can be defined. The template for some work-agent class is an object of that class with fixed itinerary that contain empty *nodes*.

All available templates are organized into a template library. A user creates a new, concrete work-agent by picking a template from the library and instantiating undefined attributes and nodes in the itinerary.

4.2. New work-agent classes

In the real companies, it is not possible to define all possible work-agents and their templates in advance. New work-agents may be needed on a daily basis. Therefore, it should be possible to create brand-new work-agents by the end-user of the workflow system. It could be done by defining and implementing a software tool that enables the visual definition of a new work-agent and its template.

4.3. Access to the Internet and databases

The suggested workflow system works only on nodes where work-servers and work-hosts are installed. In the real company environments there will be a lot of business processes that cannot be completed without access to nodes outside of the implemented workflow system. In those cases, an access to common Internet services is needed. More precisely, we need special (stationary) agents that are able to send and read e-mail messages, transfer files from one node to another using FTP protocol, access web-pages from other nodes etc.

Furthermore, work-agents will often have to access the internal or external databases, to retrieve or store data. For that purposes, a specialized database agent is needed.

4.4. Other specialized agents

The workflow system we suggest is fully distributed, without central administration, control, and maintenance. All reports, control, and management are achieved by creating and sending specialized agents that will communicate with other agents in the system and achieve the intended results. In this subsection we shortly enumerate several possibilities.

Trackers. A user of a workflow system can always send a tracker-agent. It will look for all agents owned by the user on every node in the system, and gather a report.

Detour Agents. If a user is not able for a longer period to fulfill its duties, all agents aiming to its node have to detour. A detour-agent carrying a new itinerary is sent to look for other agents and to replace their old itineraries with the new one.

Advisors. If an itinerary contains alternative routes, advisor-agents stationed at some nodes could direct incoming work-agents to take other route. They would be created automatically if some conditions are met.

5. Protection and reliability

Enforcing protection of a mobile agent system usually means: a) to protect agents from malicious nodes, and b) to protect nodes from malicious agents. Although protection is an important issue in any mobile agent system, we feel that in a closed system such is ours, all nodes can be regarded as trusted. The only way the agent communicates with foreign nodes and services, is via Internet services. Therefore, we neglect the issue of protection of agents from malicious hosts and a protection of hosts from malicious agents. This problem should be addressed when work-agents from our system are allowed for transport to foreign nodes, and when foreign agents are allowed for access to nodes of our system.

Reliability of the mobile agent system is most often regarded as enforcing the "exactly once" semantics [11]. The reliability of this kind is most commonly solved by generating spare copies of every agent ("shadows" [1] or "rear-agents").

6. Related work

Several authors have recently suggested a usage of mobile agents in workflow management (for example [2,3].) Our system is highly decentralized and consists solely of individual agents with autonomous behaviour. The only centralized control is the control of user rights to create, access, and change agents and templates.

The proposed workflow system brings some fresh views in particular fields of a workflow management and in mobile computing. In both fields, the advantages of highly decentralized and distributed approach in designing a system, have not been often recognized. Our workflow system emphasizes the fact that mobile agent has organizational advantages.

The similar holds for workflow management systems - the usage of mobile agents frees the workflow management system of centralized control that is typically the most complicated part of the system.

According to [5], most applications of mobile agents still belong to client/server software architecture. In a mobile system a client can transport itself nearer to the server-node or even to the server-node itself. The division to service providers and service users still holds however. In our workflow management system there is no such distinction. A work-agent is both a provider and a user of services.

Most agents suggested in our system (especially work-agents) are not mobile all the time. They in fact spend most of their lifetime waiting to be chosen by the user. Agents in other systems are mostly executing all the time. The need to wait on some nodes, require new features of agents. For example, externalization of agents in other systems is a useful feature, while in our system it is a key feature.

In a proposed system, conditions for agent transport to the next node are explicitly enumerated in the agent's itinerary. The creation, understanding, and maintenance of agents are therefore simpler.

7. Conclusion

The main characteristics of a workflow system suggested in this paper are almost full decentralization and distribution of workflow functions.

The proposed organization mimics usual user activities in a real flow of work. Moreover, it relieves them (or any centralized control) from the need to know what to do next with the work-agent. Every user takes care only of work-agents that are currently on its node.

Since the system consists of many autonomous agents, the system is easily changed, extended, and improved. It is often needed just to introduce new agents, without the need to change and even to understand the rest of the system.

The organization and implementation of a proposed workflow management system are also easy to understand and follow, because most of its parts are uniformly implemented as (mobile) agents. Administration and monitoring tools are implemented as special agents: trackers, detour-agents, advisors, etc. The role of workflow enactment service is done by work-servers, work-hosts, and underlying mobility of work-agents. Process definitions are embedded into templates and therefore in itineraries of individual agents.

Full implementation of the proposed system requires a lot of work on different part of the system. However, the basic system consisting of several work-agents, a work-server, and a work-host, can be achieved in a few months time. It can be immediately used, and then gradually extended with new agents. The implementation of the proposed workflow system has begun using "Aglets" - IBM's mobile agent system.

Bibliography

1. Baumann, J. And Rothermel, K., "*The Shadow Approach: An Orphan Detection Protocol for Mobile Agents*", Bericht 1998/08, Stuttgart University, Faculty of Informatics, 1998.
2. Cai, T., Gloor, P.A., Nog, S., "*Dartflow: A Workflow Management System on the Web using Transportable Agents*", Technical Report PCS-TR96-186, 1996.
3. Chang, W., Scott, C., "*Agent-based Workflow: TRP Support Environment*", Computer Networks and ISDN Systems, vol. 28, issues 7-11, 1997.
4. Debenham, J., "*Constructing an Intelligent Multi-Agent Workflow System*", Proc. of AI '98, Brisbane, Australia, pp. 119-166, 1998.
5. Gray, R. S., "*Agent Tcl: A Flexible and Secure Mobile Agent System*", PhD Thesis, Dartmouth College, Hanover, NH, SAD, 1997.
6. Harrison, C., Chess, D., and Kershenbaum, A., "*Mobile Agents: Are they a Good Idea?*", Homepage of IBM T.J. Watson Research Center, 1995.
7. Hollingsworth, D., "*Workflow Management Coalition - the Workflow Reference Model*", The Workflow Management Coalition Specification - Doc. No. TC00-1003, 1995.
8. Karnik, N.M., Tripathi, A., *Design Issues in Mobile-Agent Programming Systems*, IEEE Concurrency, July-Sept. 1998, pp. 52-61.
9. Lingnau, A. and Drobnik, O., "*An Infrastructure for Mobile Agents: Requirements and Architecture*", Homepage of Jochan Wolfgang Goethe-University, Frankfurt am Main,1997.
10. Morreale, P., "*Agents on the Move*", IEEE Spectrum, April 1998, pp. 34-41.
11. Rothermel, K. And Straßer, M., "*A Protocol for Preserving the Exactly-Once Property of Mobile Agents*", Bericht 1997/18, Stuttgart University, Faculty of Informatics, 1997.

SYSTEMOLOGY OF MULTI-AGENT DESIGN AND MANUFACTURING SYSTEMS

Georgy B. Evgenev

107005, Moscow, 2-d Baumanskaya st. 5, BMSTU, Russia
egb@sprut.bmstu.ru
tel (095) 263-69-70

Abstract

Implementation of multi-agent systems for design and manufacturing is described. Characteristics of such systems are involved and discussed. Models of agents, multi-agent systems and their environment are presented..

Keywords: *multi-agent systems, distributed intelligence, knowledge base, CAD, CAM.*

1. Introduction

Design and manufacturing is a very fruitful field for multi-agent system methodology implementation. In this domain the main goals is reducing time and cost of those processes. To achieve the goals it is necessary to develop information technology of creation integrated intelligent CAD, CAPP and CAM system, which have to become the means of concurrent engineering.

According to [1] "an agent is a real or virtual entity which emerges in an environment where it can take some actions, which is able to communicate with the other agents and which possesses an autonomous behaviour that is consequence of its observations, its knowledge and its interactions with the other agents".

An agent is a problem solving system operating within a loosely coupled network of other agents (MAS) that work together to solve problems which they are not able solve on their own [2].

Within the domain of multi-agent design and manufacturing systems (MADMS) agents are virtual entities. Environment, where agents take actions, is a project, which has to be completed and which data are used as a mean of inter-agents communication. Each agent has its own properties and knowledge of functional dependencies between those properties. This knowledge defines the behaviour of an agent. The goal of any agent is to determine its own properties using data of project state observations along with structure and demanded input properties of depended agents. So agents work together to complete a project which incorporate the data of a set of agents examples.

Engineers are needed the very specific agents. Among them are assemblies, subassemblies, parts and its elements along with manufacturing routing processes, operations, suboperations and so on.

2. Systemology and MADMS Architecture

There are two different types of knowledge representation: procedural and declarative. Algorithms are a mean of procedural representation but models — declarative one. Within Artificial Intelligence are used models.

Any model includes two basic components: set of the objects and set of the relations.

$$M = < A; R_1^{s1}, \dots , R_n^{sm} >$$

Here M — model, A — carrier of model, R_i^{si} — i- relation ($i = 1, \dots , n$), which set forms signature of model. If to use as criterion of classification a type of the carrier, one can obtain the traditional division of a science, any branch of which is engaged in investigation of the certain type of elements (physics, chemistry, mechanical engineering, electronics etc.). Completely other classification of systems one can develop using the relations as criterion. The science studying systems in this aspect, is systemology [3]. From the systemic point of view the hierarchy of classes define epistemological levels, i.e. levels of knowledge

The lowermost level in this hierarchy designated, as a 0-level is system, recognised by the researcher as such. At this level the system is defined through the set of properties and carries the name initial system. Other words at a 0-level there is considered properties of researched or projected system.

On higher epistemological levels systems differ each other by level of knowledge of the variables appropriate initial system. In systems on higher level there are used all knowledge of systems of lower levels and besides the additional knowledge which inaccessible to the lowest levels.

After the initial system is complemented by the data, i.e. actual variables values, there is involved a new system. This is an initial system with the data, which dispose on the 1-st epistemological level. The systems of this level refer to as data systems.

The 2-nd level represents a level of knowledge bases for generation of variables values, determining properties of initial system. At this level the functional relations of variables are set. Variables include ones, determined by the appropriate initial system and, probably, some additional. As a main task of this level is a generation of the initial system properties, the systems of the 2-nd level refer to as generative systems.

On the 3-rd epistemological level systems determined as generative or system of a lower levels, refer to as subsystems of common system. These subsystems can incorporate in the sense that they have some common variables. The systems of this level are named the structured systems.

On the 4-th epistemological level and higher levels the systems consist of a set of systems determined at the levels 0, 1, 2 or 3, and some metacharacteristics (rule, relation, procedure), describing replacements in systems of a lower levels. There are levels necessary for formation conceptual AND - OR graphs.

An overall architecture of MADMS comprises a pair: environment and multi-agent system.

$$MADMS = <E, MAS>$$

MADMS — the multi-agent CAD/CAM; *E* — the environment (project conceptual database); *MAS* — the multi-agent system.

Environment is a project conceptual database. Models of its elements are disposed on the first two knowledge levels: initial systems and data systems. The whole environment is a structured system comprised from those elements.

Agent is a pair of initial system, which define its properties, and structured generative system, which defines its method.

The model of multi-agent system is a metasystem disposed on the highest knowledge levels.

3. Model of MADMS Environment

There are two main classes of multi-agent environments: non-transformable and transformable. Both classes have two subclasses: closed and open [4]. Transformable environments can change their descriptions as a result of agent's actions. Knowledge about such environments and its descriptions has a temporal type. Changes of state of transformable environments are planned and initiated by agents. MADMS environment belongs to transformable class.

The closed environment provides exhaustive description of itself. Agents can take information about its state in process of interactions with environment. The goal of MADMS is to gain exhaustive project description, so its environment has to belong to closed subclass of transformable class.

The environment can be deterministic or probabilistic [4]. MADMS works with deterministic closed transformable environment.

In case of MADMS, it is very useful to detach an invariant kernel from environment. This kernel has to incorporate standard and common used conceptions and its attributes. Kernel concepts are connected closely to drafting standards such as ANSI, DIN, ESKD etc.

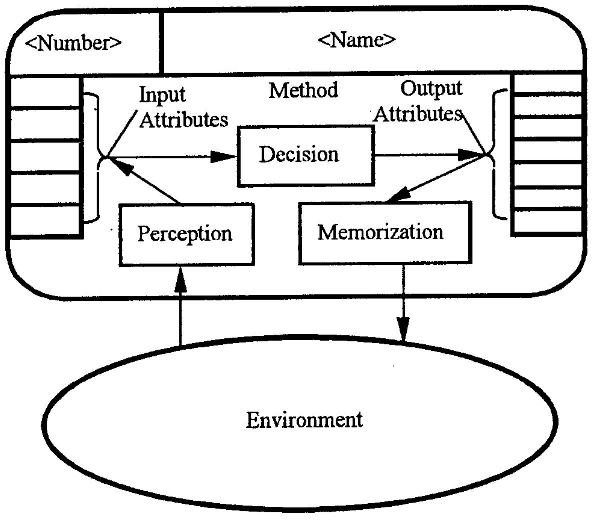
The kernel includes such basic concepts as "Item" and its different kinds: standard, purchased unified and novel assemblies and parts. Composition of assemblies is defined by specification. Joints with its different types (welded, threaded and so on) describe how assemblies are making up. Dimensions of items and its accuracy are defined by appropriate concepts. Concepts "Material", "Coating", "Roughness", "Hardness" describe properties of parts.

The environment kernel includes also such concepts as "Manufacturing plan", routing assembly and part manufacturing processes and its operations.

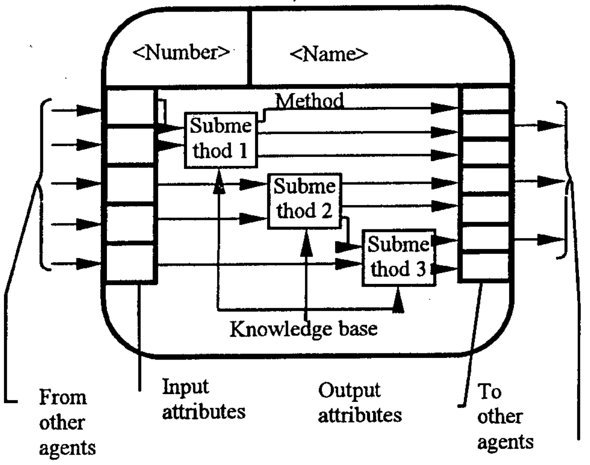
The implementation of MADMS in various manufacturing domains is fulfilled by including in environment concepts of appropriate domain (for example: "Drive", "Electro-engine", "Muff", "Reducer" etc.)

4. Agents Model

There are two different types of agents: agent and subagent (fig.1). The structure of agent method includes three blocks: perception, decision and memorisation [5]. The perception block makes agent to be able to perceive environment conditions. When data related to agent emerge in the environment, it becomes active, read out this data, make appropriate decisions and transform environment through memorisation block.



a)



b)

Fig. 1. Architecture of agent (a) and subagent (b)

Subagent is a component of complex agent, which can be regarded as super-agent. Method of subagent has not perception and memorisation blocks and so subagent can not transform environment. Subagents are interchanged by data directly within decision block of super-agent.

Each MADMS agent is connected closely with one of the environment concept. Therefore agent is a pair of concept schema and method.

$$Ag=<shm\ P, M>$$

Ag — the engineering agent; *shm P* — the schema of notion; *M* —the method of agent.

Concept schema includes three sets of attributes: characteristic, valent and differential.

$$shm\ P = <B, C, D>$$

$B=\{B_j\}, j=1,..., q$ —the set of characteristic attributes;
$C=\{C_k\}, k=1,..., m$ —the set of valent attributes;
$D=\{D_l\}, l=1,..., n$ —the set of differential attributes.
By characteristic attributes there is accomplished a distinguishing of diverse agents by its names and examples of each agent by it number.
Differential attributes define agent properties. They are divided on two classes: input and output.
Valent attributes are needed to describe connections between different agents.
Agent method is a pair of production rules set and semantic network of those rules [6]

$$M=<PR, N>$$

PR— the set of production rules; N —the semantic net of production rules.

5. MAS Model

Within the scope of knowledge engineering multi-agent system has a structure, which is defined by AND/OR graph.

$$MAS = <Ag, R>$$

MAS — the multi-agent engineering system; Ag — the set of engineering agents; R — the inter-agent connections:

$$R \subseteq Ag \times Ag \times Con$$
$$Con = \{AND, OR\}$$

MAS structure depicts using notations of IDEF0 and IDEF1X standards. IDEF0 is used for functional analysis of item
Within MADMS multi-agent structure has not explicit representation. Each agent has its own knowledge about local fragment of overall AND/OR graph.

5. Characteristics of MADMS

In domain of design and manufacturing is used reactive or plant-like category of agents' [7]. Reactive agents are the simplest kind of ones but as it was proved by practice such agents can decide all problems in this domain. It is a further development of the expert systems consisting of a knowledge base build out of rules, a fact base, and an inference machine. Within of MADMS this system gain capability to communicate with other agents in order to decide a common problem.
MADMS agents can be regarded as "actors" within "actor system"[7]. The actions of such actors as the actions of the system as a whole get their meanings by means of a message passing semantics, i.e. a semantics which ab initio is based on possible messages that can be exchanged by the individual actors. Knowledge base of each actor has to understand the sense of messages that are sent to it by other actors via environment. This function is fulfilled by perception block of agent method.
As a result of mentioned above agent properties, MADMS agent's method development is based on production rules of knowledge base.
MADMS agents use the both types of communication mode: direct and indirect. Subagents within of super-agent use direct mode. Usual type of inter-agent communication is indirect one via environment.
As it has been discussed above, MADMS environment belongs to transformable closed class and agents communicate each other through this environment by sending of semantic messages.
Within MADMS the most important agents are human beings. So MADMS has to use advanced human-computer interaction (HCI) on the base of business prose language. In the most cases human operator is involved into decision making and is an "actor" among other artificial "actors". In those cases are used "participant system" mode of HCI.
MADMS are used initiative activation of agents but within the super-agents activation belong to imperative type.
The most kind of agents' distribution in MADMS is distributed network but at the small enterprises agents can be located on the one workstation. The platform of distributed MADMS is a net of specialised workstations so there is not need to use of agent's migration.
When the designing is based on the invariant item structure and is confined by parametrical synthesis, multi-agent system structure has a static kind. Otherwise this structure is a dynamic one.

6. Conclusion

Technology of multi-agent systems is a most promising information technology for computer integrated manufacturing in 21st century. MADMS is a mean of achievement of the main conceptual goals in this domain: integration, intellectualization and individualization. On the foundation of described above concepts there was developed instrumental workbench of multi-agent oriented programming for design and manufacturing.

Bibliography

1. Moulin, B. and Chaib-Draa, B. An Overview of Distributed Artificial Intelligence. In: *Foundations of Distributed Artificial Intelligence* (Eds. O'Hare, G.M.P. and Jennings, N.R.), Chapter 1, Jon Wiley & Sons, Inc., New York, 1996.
2. Durfee, E.H., Lesser, V.R., and Corkill, D.D.. Trends in cooperative distributed problem solving, *IEEE Trans. Knowl. Data Eng.* KOE-11(1), 63-83, 1989.
3. Klir, G.J. Architecture of systems problem solving. *Plenum Press*, New York, London, 1985.
4. Pospelov D.A. From Collective of Automates to Multi-agent Systems. *Proc. of the Intrnational Workshop "Distributed Artificial Intelligence and Multi-Agent Systems" DAIMAS'97*, St.Petersburg, Russia, 319-325, 1997 (in Russian)
5. Ferber J., Labbani O., Muller J.-P., Bourjault A. Fromalising emergent collective behaviours: preliminary report. *Proc. of the Intrnational Workshop "Distributed Artificial Intelligence and Multi-Agent Systems" DAIMAS'97*, St.Petersburg, Russia, 113-123, 1997
6. Evgenev G.B. Multiagent methodology for computer aided design and integrated manufacturing, in: *Globalization of Manufacturing in the Digital Communications Era of the 21st Century: Innovation, Agility, and the Virtual Enterprise. Proceedings of the Tenth International IFIP WG5.2/5.3 International Conference PROLAMAT 98*, Trento, ITALY, 591-602, 1998.
7. Braspenning P.J. Plant-like, Animal-like and Humanoid Agents and Corresponding Multi-Agent Systems. *Proc. of the Intrnational Workshop "Distributed Artificial Intelligence and Multi-Agent Systems" DAIMAS'97*, St.Petersburg, Russia, 64-77, 1997

STUDY OF BEHAVIOUR OF A MULTI-AGENT ENVIRONMENT WITH INTELLIGENT DATA BASE AGENTS

Mieczyslaw A. Klopotek[1], Tomasz Nowak[2]

[1]A Institute of Computer Science PAS, Ordona 21, 01-237 Warsaw, Poland
[2]Apolish Telecommunication SA, Warsaw, Poland.

Abstract
The paper presents a multi-agent system with agents cooperating via knowledge exchange on resolving local queries to databases. The system permits to audit the behavior of a population of cooperating agents when varying strategies of agents. As a behavioral background operation of home shopping agencies has been selected. The system is implemented in Java. Agents communicate using a language similar to KQML].

Keywords: *multi-agent systems, simulation, distributed intelligent databases*

1. Introduction

Multiagent systems consist of many relatively small programs - àgents. Each of them pursues a goal of its own, can communicate with the other ones and can carry out various actions in the system. [3,4] The advantage of programming using the method of multi-agent systems is that there is no need to know the global solution of the problem under consideration and the environment in which the system is running can be changing and heterogeneous. The other advantage is the universalism of the agents. An agent written once can act in various environments.

The paper presents a multiagent system with agents co-operating via knowledge exchange [5,6] on resolving local queries to databases [2]. The system permits to audit the behavior of a population of cooperating agents when varying strategies of agents. As a behavioral background operation of home shopping agencies has been selected. The system is implemented in Java [1], Agents communicate using a language similar to KQML [9]. In section 2 the home-shopping problem considered is briefly outlined. Section 3 describes the simulation environment for the problem. Section 4 gives the results of our experiments with home-shopiing agent strategies. We conclude the paper with short remarks in section 5.

2. Home Shopping Problem

Most of us encountered forms of type "Home-Shopping" (HS). Their activity consists in sending advertisement leaflets to homes of potential clients. They offer a variety of more or less useful products. The client that received such a leaflet can select and order the advertised products.

The process may clearly be automated in that a database of addresses and formalized properties of potential clients is maintained by a database (DB) agent, and is used by a service (US) agent in order to find potential buyers satisfying specification of a seller (client) of a product. Usually, a US agent operates in a certain (geographical) region and competes with other US agents in the same region and may cooperate with US agents operating in other regions (e.g. by exchanging information on buyers' behavior). A US agent works then as follows (Fig. 1)

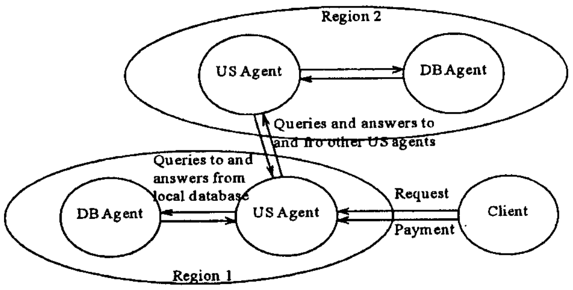

Region 2
US Agent
DB Agent
Queries and answers to and fro other US agents
Queries to and answers from local database
Request
Payment
DB Agent
US Agent
Client
Region 1

Fig. 1 How the US Agent meets the request of the client.

1. The client passes its request to the US Agent
2. The US Agent communicates with his DB Agent, checking if all attributes appearing in the client's request are also available in his data. If so, the asks the DB Agent to select all inhabitants matching client's criterion and sends them the offer. If some attributes are missing, he asks other US Agents if they are able to calculate the dependence between the attributes he knows and those missing in the request of the client. If the reply is positive, he substitutes the unknown attributes with proper expressions and requests the DB Agent to select inhabitants according to the modified criterion. He has also the choice to request the DB Agent to collect the missing data in the region. However, maintaining all possible data is not economical, because buying and maintaining attributes must be paid for. On the other hand inhabitants of different regions differ in their characteristics so that dependencies derived from other regions may be prune to errors. Hence one should use information from regions that are similar.
3. The client, after getting the reply to its offer from the inhabitants pays the US agent for realization of its request.

3. Simulation Environment

The environment consists of four main components (programs) [8]

- Database Agent integrated with database server
- US Agent Frame, permitting to define agent's strategies
- Automatic client - user
- Monitor auditing the agent's state

Database Agent.

The whole system is a distributed database consisting of many non-overlapping local databases. Each local database is administrated by one database agent (DB Agent). His functions are:

- Processing queries to the database
- It enables creation of new attributes
- It calculates functional relations between attributes.

The agent collects also fees from US agents using its services. It is paid for creating a new attribute and for maintaining attributes. Maintaining must be paid for regularly independent of usage frequency of the attribute.Queries to the database are formulated in a simplified version of the SQL language. An answer to an SQL query is a group of rows from the database formatted as the list of columns after the keyword 'select':
Column1 Column2 ...
Queries for knowledge (dependencies) are of the form:
Column(Column1, Column2, ...)
where *Column* is the sought attribute
Column1, Column2, ... - known attributes
The answer is:
Column(Column1, Column2, ...) ::= Expression
where *Column* is found attribute
Column1, Column2, ... - attributes on which the attribute *Column* depends

US Agent Frame.

The task of the Utility Agent (US Agent) is to realize client's queries. The HS client poses a question to its US Agent. It is the US Agent that needs to bother where to find data to match client's request. US Agent can communicate with its database agent and other US agents. Replies of the database agent are exact, that is they represent concrete lines from the database. On the other hand, the other US agent may provide with functional relationship between attributes. The US agent has an initial account state. He has to pay for maintaining columns in the database and for functional dependencies from other US agents. He earns money providing other US Agents with functional dependencies and realizing HS client requests.

The frame is the agent's interface with the environment. While creating a new agent we can concentrate only on the strategic aspects of the agent's algorithms with no need to program communication protocols with the environment. The frame imposes a concrete structure of the US Agent making creation of an agent a relatively simple task.

Automatic client.

To evaluate the long term performance of US Agent, he must be submitted with many requests. The stream of requests is generated by an automatic client. One can specify a number of request types and specify relative frequencies of these requests. The client will then send them randomly.

Monitor.

It is an application permitting to audit operation of US Agents. One can view the states of all US Agents in the system that is their accounts and columns bought in the database.

4. An Experiment

To test the environment, a database has been generated and two US agents have been created. The resulting database consists of 9000 rows (1000 rows for each region). The first of the agents has been called „First». And works according to the following scheme.

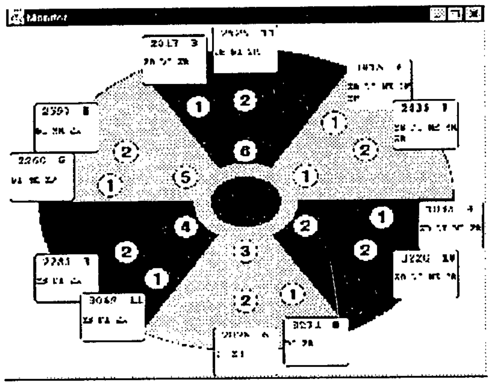


Fig. 2- Results of agents of type „First»'

```
after _receiving _a _request _from _the _client()
    check _if _all _attributes _needed _are available _to _you()
    if yes then realize _the _request();
        answer _to _the _client();
    if no then for each _missing _attribute
            if dependence _has _already _been _bought then buy _the _dependence();
            if dependence _has _not _been
            _bought _earlier or attempt _to
            _buy _failed then
            buy _the _attribute();
        for each _bought _attribute
            ask _other _agents _about _dependence ();
            if dependence found and accuracy _of _dependence>75%
                resign _from _the _attribute();
                remember _by _whom _found ();
        substitute _the _dependence _into _the _request ();
        realize _the _request();
        answer _to _the _client ();
```

The second agent is called „second». Its algorithm is simpler

```
after _receiving _a _request _from _the _client()
        check _if _all _attributes _needed _are available _to _you()
        if not then
                for each _missing _attribute
                        buy _the _attribute();
                realize _the _request();
                answer _to _the _client ();
```

The algorithm of the agent „Second» consists in buying and maintaining all columns in the database that were ever needed to reply to client's requests. As maintaining columns is expensive, after some time this strategy leads to bankruptcy.

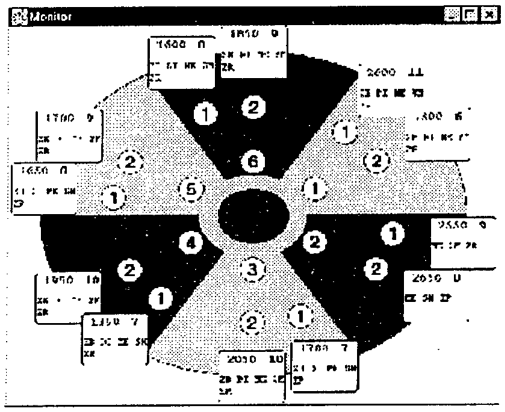


Fig. 3 - Results of agents of type „Second»'

The algorithm of the agent 'First' is cost-saving.. At the beginning it is buying all columns needed. But during further operation it tries to reduce the number of columns maintained. It asks questions to other agents to check accuracy of dependencies derived from their data. If one with sufficient accuracy is identified, then the agent resigns from maintaining the column. As obtaining a dependence is cheaper than maintaining a column, so the agent „First» works more effectively than the „Second». Additional effect is the specialization of agents. Some maintain all columns but earn by replying to queries about dependencies. The other reduce the number o columns and make extensive use of dependencies. the automatic client issued the following requests with identical probabilities:
select TOYS, CHILDREN, INCOME from MAS where TOYS > 500;
select CAR, INCOME from MAS where CAR>0;
select FLAT, INCOME from MAS where FLAT > 150;

Two experiments were run. In the first experiment agents of type „First» were run. In the second - agents of type „Second». After several „rounds» the states visible in Fig.2 and Fig. 3 resp. were arrived by respective agent populations. The figures make visible (for description of the meaning of information content of Monitor program window see Fig 2.), that agents of type „First» survive and the state of accounts grow (the initial state was 2000). Most of agents of type „Second» get account balance below the initial value. Also specialization of agents of type „First» become visible. Agents in region 1 maintain all columns but earn thanks to responding to queries about dependencies. Agents in region 2 do not maintain one column. One can conclude that they partially make use of dependencies provided by agents from region 1 and partially earn money from providing functional dependencies. The remaining agents maintain only three attributes and they recover the missing information receiving functional dependencies from agents from regions 1 and 2.
Agents of type „Second» increase the balance of their accounts in the first stage of simulation. The reason is the low initial costs. As long as the clients' queries concern only a small number of attributes., the maintaining of attributes costs less by incomes from request answering. However, later the situation changes and the costs of maintaining all columns are higher than incomes.
In case of agents of type „First» the situation is different. Initial costs are high because simultaneously they buy the needed columns and ask other agents about dependencies to check their accuracy. Later costs fall because after finding proper cooperating agents . the agents themselves can resign from maintaining superfluous columns.

5. Concluding Remarks

As a behavioral background operation of home shopping agencies has been selected and an experiment comparing two different strategies of agent's operation have been tested. As expected, more elaborate strategies consisting in cooperative usage of knowledge of other agents, though more expensive at the beginning, are more cost effective in a long term.

Bibliography

1. Davis S. R. : Learn Java Now, Microsoft Press, 1996
2. Goczylak : Architektury współczesnych systemów baz danych, INFOBAZY '97, Proceedings
3. Katarzyniak R.: Zastosowanie autonomicznych programów agenckich do wyszukiwania informacji w heterogenicznych rozproszonych bazach danych, INFOBAZY '97, Proc.
4. Nwana H. S.: Software Agents, Knowledge Engineering Review, Vol. No 3, pp. 1-40, Sept 1996.
5. Ras Z.W.: Collaboration Control in Distributed Knowledge-Based Systems, „Information Sciences Journal", Elsevier, Vol. 96, No 3/4, pp 193-205
6. Ras Z.W.: Cooperative Knowledge-Based System, „Intelligent Automation and Soft Computing Journal", Vol. 2, No 2, 1996, pp 193-202
7. Klopotek M.A., Nowak T. : An environment for simulation investigations of behaviour of a population of home-shoppimg agents. In: Proc. IIS'98, Malbork, Poland, June 15-19, 1998, Publisher IPI-PAN, pp. 194-198.
8. Nowak T., Klopotek M.A.: Srodowisko do badan symulacyjnych nad zachowaniem sie populacji agentow typu home-shoppping. Proc. 3rd National Conference on Artificial Intelligence CIR-13'98, Siedlce, 16-17.9.1998 . Publisher ZG PTC/II WSRP Siedlce, pp. 299-304.
9. Mayfield J., Lebron Y., Finin T.: Evaluation of KQML as an agent communication language. Intelligent Agents II. Lecture Notes in Artificial Intelligence 1037

THE PROPOSITIONAL N-AGENT LOGIC

Nickolai K. Kossovski[1], Artem V. Tishkov[2], Vladimir V. Iaroslavski[3]

[1] *PhD, The Head of Computer Science Department, Saint-Petersburg State University; h. 2, Bibliotechnaya pl., Mathematics and Mechanics Faculty, Petergof, Saint-Petersburg, 198904; kosov@nkk.usr.pu.ru, phone (812) 155-6903.*

[2] *PhD-student of Computer Science Department, Saint-Petersburg State University; h. 2, Bibliotechnaya pl., Mathematics and Mechanics Faculty, Petergof, Saint-Petersburg, 198904; Artem.Tishkov@paloma.spbu.ru.*

[3] *PhD-student of Computer Science Department, Saint-Petersburg State University; h. 2, Bibliotechnaya pl., Mathematics and Mechanics Faculty, Petergof, Saint-Petersburg, 198904; rpetrodvoretc@gov.spb.ru, vlv@math.spbu.ru, phone (812) 427-5649.*

Abstract

The propositional n-agent logic is proposed. The graphical illustration is given. An example is considered. A sequent calculus for the n-agent logic is introduced. It is proved that the decision problem of propositional n-agent logic belongs to EXPTIME.

Keywords: *heuristic logic, agents, algorithm complexity.*

1. Introduction

Multi-agent technology is developed in [12]. One can use his own logic: either two-valued logic or three-valued logic for the same facts. The n-agent logic allows to unify the assertions and reasoning to one system.

R.S. Michalski [11], [1] has proposed a variable-valued logic system as a form of union of all Post [14] logics together with fuzzy logic introduced by L.A. Zadeh [15].

This paper is a further development of [6]–[8]. We propose the n-agent logic where the n-agent logical value is a sequence of n voices or teams. Each team uses the logic from [7] with logical values and connectives easily representable on a computer.

The introduced system is able to replace fuzzy and continuous logics [13], [15] since in practice only rational numbers are used.

Some logic which generalizes all Post logics was introduced in [4]. Another approach based on interval logic is presented in [9]. Probabilistic ideas are used in [5]. Other many-valued logics are described in [2], [3].

In the proposed n-agent logic we can receive the same results in mathematical psychology concerning to human consciousness as gamma-algebra of Lefebvre, introduced for subconscious (intuitive) processes of human thinking simulation [10].

2. The n-agent logic language

Let n be a natural number. The n-agent logical value is an ordered sequence of n rational numbers. The notation $[A]$ will be used to denote the value of a formula A. The *average value* of P such that $[P] = (p_1, \dots, p_n)$ is denoted as $avg(P)$ and is defined as

$$avg(P) = \frac{1}{n}\sum_{i=1}^{n} p_i.$$

We define a *hyperplane of contradictions* in $\mathbf{Q}^n$ as a set of *contradictory points*, i.e.

$$\{(p_1, \dots, p_n) \in \mathbf{Q}^n \mid p_1 + \dots + p_n = 0\}$$

and define an *axis of truth* in $\mathbf{Q}^n$ as a set of points

$$\{(p_1, \dots, p_n) \in \mathbf{Q}^n \mid p_1 = \dots = p_n\}.$$

Notice that the value $\left|\sum_{i=1}^{n} p_i\right| / \sqrt{n}$ is a distance between the point $P = (p_1, \dots, p_n)$ and the hyperplane of contradictions.

Every value which is above the hyperplane of contradictions is ***true in average***, every value which is under the hyperplane of contradictions is ***false in average***.

The propositional formula of the n-agent logic is constructed of propositional variables and logical connectives as usually. Let P and Q are the n-agent propositional formulae and

$$[P] = (p_1, \dots, p_n), \quad [Q] = (q_1, \dots, q_n).$$

The "n-agent negation" ($\urcorner P$) is represented by a symmetry over the hyperplane of contradictions and is defined by the formula

$$[\urcorner P] = [P] - 2\,Avg(P)$$

where

$$Avg(P) = \left(\frac{1}{n}\sum_{i=1}^{n} p_i, \dots, \frac{1}{n}\sum_{i=1}^{n} p_i\right).$$

The "logical negation" ($\neg P$) is the negation of all coordinates: $[\neg P] = -[P]$. The n-agent operation "symmetry over" ($\dagger P$) is represented by a symmetry over the axis of truth and is defined by the formula $[\dagger P] = [\neg \urcorner P]$.

The n-agent disjunction is defined by

$$[P \hat{\vee} Q] = \begin{cases} [P], & \text{if } avg(P) > avg(Q), \\ [P], & \text{if } avg(P) = avg(Q) \text{ and } [P] <_L [Q], \\ [Q], & \text{otherwise,} \end{cases}$$

where relation "$<_L$" is a lexicographic order.

The n-agent conjunction is defined by

$$[P\,\hat{\&}Q] = \begin{cases} [P], & \text{if } avg(P) < avg(Q), \\ [P], & \text{if } avg(P) = avg(Q) \text{ and } [P] <_L [Q], \\ [Q], & \text{otherwise.} \end{cases}$$

The n-agent "strengthening" and the n-agent "weakening" are defined by $[P!] = 2[P]$ and $[P?] = \frac{1}{2}[P]$ respectively.

Notice that De Morgan's laws are fulfilled:

$$^{\neg}(P\,\hat{\&}\,Q) = {}^{\neg}P\,\hat{\vee}\,{}^{\neg}Q, \quad ^{\neg}(P\,\hat{\vee}\,Q) = {}^{\neg}P\,\hat{\&}\,{}^{\neg}Q.$$

The value of the n-agent formula P may be considered as assertions of n agents. The average value of P may be considered as the average opinion of all agents.

The generalization of the n-agent operations is available. It is possible to introduce the "$(i_1, \ldots, i_m)$-negation". This operation is considered as a symmetry over the hyperplane $p_{i_1} + \cdots + p_{i_m} = 0$ where $1 \leq i_1 < \ldots < i_m \leq n$ and $m < n$.

3. Example *(Election in Russian Federation)*

The n-agent technology may be useful for the analysis of an election result. Let consider the election to the Legislative Assembly of Saint-Petersburg which takes place every four years. The area of Saint-Petersburg is divided into 50 election districts. So fifty deputies are elected to the Legislative Assembly of Saint-Petersburg. There are some candidates for deputy from each election districts. The ballot-paper consists of candidate names and item "against of all candidates" at the end. The result of election depends on many factors. For example, a winner is not elected if the percentage of votes is less than 25%, or if a number of votes for a winner is less than a number of votes "against of all candidates".

Therefore, we can consider a value p_i from $[P]$ as a number of votes for the i-th candidate where $i \in [1, n-1]$ and the value of p_n as a number of votes "against of all candidates" multiplied by (-1) or by $(-n+1)$. If we take the coefficient $(-n+1)$ then $[P]$ was true in average in some districts in the election to the local region administration of Petershof (1997) and to the Legislative Assembly of Saint-Petersburg (1998, round I). So we can use "coefficients" for more convenient knowledge representations.

4. Heuristic logic

Heuristic logic was proposed in [6]. Let us consider 2-agent knowledge representation for expert systems. Truth of assertion is represented by an ordered pair (main and secondary, sometimes opposite, meanings). It is based on heuristic logic. The value of a heuristic formula is an ordered pair of two rational numbers.

Let P and Q be heuristic formulae and their values are $[P] = (p_1, p_2)$ and $[Q] = (q_1, q_2)$. The heuristic operations "&" and "$\vee$" are interpreted as

$$[P \vee Q] = (\max(p_1, q_1), \max(p_2, q_2))$$

and

$$[P \,\&\, Q] = (\min(p_1, q_1), \min(p_2, q_2)).$$

The operation "heuristic negation" is interpreted as $^{\neg}P = (-p_2, -p_1)$. The logic negation is interpreted as $[\neg P] = (-p_1, -p_2)$.

Combination of operations "$\neg{}^{\neg}$" is heuristic operation "on the other hand" ($\backslash P$) and is interpreted as $[\backslash P] = (p_2, p_1)$. The heuristic operations "strengthening" ($P!$) and "weakening" ($P?$) are interpreted as $[P!] = (2p_1, 2p_2)$ and $[P?] = (p_1/2, p_2/2)$ respectively. Consider the graphical illustration of heuristic logic on the fig. 1.

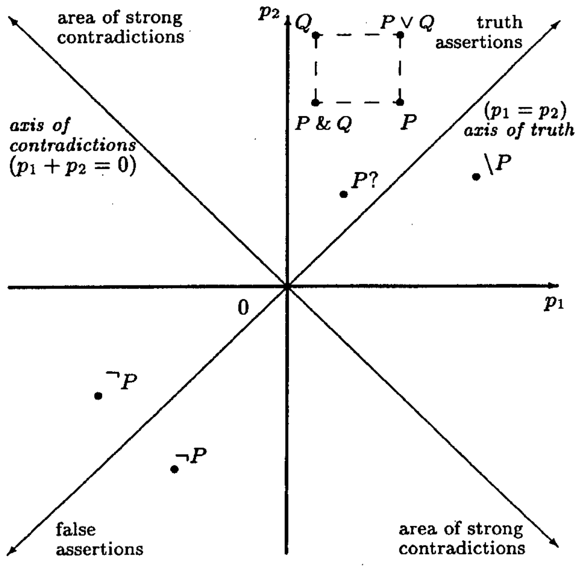


Fig. 1. Graphical illustration of heuristic logic

The operation of logic negation $\neg P$ is a symmetry over the origin point O. The operation "&" is graphically interpreted as a left bottom corner of the rectangle formed by points P and Q. The operation "$\vee$" is interpreted as a right top corner of the same rectangle. The operation "heuristic negation" ($^{\neg}P$) is a symmetry over the axis of contradiction. The heuristic operation "on the other hand" ($\backslash P$) is a symmetry over the axis of truth. So $\backslash P = \uparrow P$. The heuristic operations "weakening" ($P?$) and "strengthening" ($P!$) are interpreted as shift of point P along the axis OP.

The n-agent logic is a generalization of heuristic logic. Another generalization was announced in [8].

5. Main result

Theorem 1 *Decision problem of propositional n-agent logic belongs to EXPTIME.*

Proof of this theorem is based on Gentzen-style sequent calculus.

Let T be a signature. Under a quantifier-free theory based on T and denoted by $UTh(\mathrm{T})$ we mean a set of quantifier-free predicate formulae of classical first-order logic such that interpretations of their (universal) closures are true under ordinary interpretation of elements of T.

Variables of a formula have indexes. Coefficients and indexes are written in the binary notation. The set of constants may be included as functions without arguments. A semicolon is written in a signature before the function and predicate list.

Define a signature for quantifier-free theory $\overline{UQ}^n$:

$$UTh(\mathbf{Q^n}; |\mathbf{Q^n}, |\{\lambda x.c \cdot x + c_0 : c \in \mathbf{Q}, c_0 \in \mathbf{Q^n}\},$$
$$\lambda x.abs(x), \lambda x. \setminus x, \lambda ab.a + b, \lambda a.(0 \leq a))$$

where $\mathbf{Q^n}$ is the set (and $|\mathbf{Q^n}$ is a sequence) of systems of n rational numbers, $|\{\lambda x.c \cdot x + c_0\}$, $\lambda x.abs(x)$, $\lambda ab.a + b$ are usual linear, absolute value and addition functions, $\lambda x. \setminus x$ is a cyclic coordinate permutation and predicate $\lambda a.(0 \leq a)$ is a comparison of the first coordinate with the zero.

A sequent is a word which begins with the sequent sign $\rightarrow$ followed by a list of quantifier-free formulae of two-valued logic using the only predicate of inequality. An interpretation of a sequent $\rightarrow \Delta$ is a disjunction of all predicate formulae in Δ. An interpretation of the sequent with the empty list Δ is false.

A base of a sequent S is a sequent which is obtained from S by removing all its formulae except of comparisons with the zero of linear combinations of individual variables and constants. An axiom is a sequent with a true base, i.e. the base contains at least one true inequality after substitution of any set of values instead of variables.

This definition is equivalent to the following assertion. The system of negations of inequalities of axiom's base is unsolvable. The notions of derivation, derivable formula and derivable sequent are usual. A derivation of a sequent S is a sequence of sequents such that each sequent in it is either an axiom or has been derived from previous sequents by some rule and S is the last sequent of the derivation.

Lemma 1 *Interpretations of axioms are true and interpretations of inference rules' conclusions are true if interpretations of their premises are true.*

Lemma 2 *Every sequent which belongs to $\overline{UQ}^n$ is derivable.*

Axiom recognition algorithm consists in unsolvability checking for a system of rational linear inequalities. It is well-known that this problem is polynomial.

One can reduce the decision problem for the n-agent logic to the decision problem for $\overline{UQ}^n$, keeping the length of $\overline{UQ}^n$ formula polynomial in the length of the n-agent formula, and so we get EXPTIME upper bound for decision algorithm for the n-agent logic.

Bibliography

[1] R. Chilavsky, B. Jacobson, R.S. Michalski "An application of variable-valued logic to inductive learning of plant disease diagnostic rules". Proceedings of the sixth international symposium on multiple-valued logic. — Utah State University, Logan, Utah, 1976. — pp. 233–240.

[2] D. Dubois, H. Prade "Fuzzy sets in approximate reasoning". Part 1: Inference with possibility distributions, Fuzzy sets and systems, V.40, № 1. — Amsterdam: North-Holland, 1991. — pp. 143–202.

[3] D. Dubois, J. Lang, H. Prade "Fuzzy sets in approximate reasoning". Part 2: Logical approaches, Fuzzy sets and systems, V. 40, № 1. — Amsterdam: North-Holland, 1991. — pp. 203-244.

[4] A. Getmanova "Logic". — Moscow: Progress Publishers, 1989 (in Russian).

[5] J.Y. Halpern, M.O. Rabin "A Logic to Reason about Likelihood". Artificial Intelligence, 32, 1987. — pp. 379–405.

[6] N.K. Kossovski "Derivation of comparisons of logical formulae estimates". Proceedings of II International conference "Discrete models in the control systems theory". — Moscow, 1997. — pp. 33–35 (in Russian).

[7] N.K. Kossovski, A.V. Tishkov "Mathematical reasoning for fuzzy propositions". Proceedings of International Conference on Informatics and Control. — Saint-Petersburg, 1997. — pp. 522–529.

[8] N.K. Kossovski, A.V. Tishkov "Derivability in a pluralistic logic". Modern logic: problems in theory, history and applications in science. — Saint-Petersburg, 1998. — pp. 151–154 (in Russian).

[9] O. Kosheleva, V. Krenovich "One More Potential Application of Symbolic Interval Logic". Interval 94, Abstracts. International Conference on Interval and Computer Algebraic Methods in Science and Engineering. — Saint-Petersburg, 1994. — pp. 143–146.

[10] V.A. Lefebvre "An outline of Fundamental Psychology". School of Social Sciences. — University of California, Irvine, 1991.

[11] R.S. Michalski "Variable-Valued Logic: System VL_1". International Simposium on Multiple-Valued Logic. — West Virginia University, Morgantow, 1974.

[12] H.S. Nwana, D.T. Ndumu "An introduction to agent technology".
Advance Application & Technology Department, UK. Software agents and soft computing. ISBN 3-540-62560-7, Springer-Verlag, Berlin, Germany, 1997. — pp. 3–26.

[13] L.I. Volgin, V.I. Levin "Continuous logics, theory and applications". — Tallinn, 1991 (in Russian).

[14] E.L. Post "Introduction to a General Theory of Elementary Proposition". American Journal of Mathematics, vol. 43, № 3, 1921. — pp. 163–185.

[15] L.A. Zadeh "Fuzzy Logic". Neural Network and Soft Computing. Communications of the ACM, 37, 1994. — pp. 77–89.

USE OF A VISUAL BLACKBOARD MULTIAGENT SYSTEM IN MOULD INDUSTRY

Jesús Lorés[1], Joan Vivancos[2], Xavier Sirera[3]

[1]Computing Department, Universitat de Lleida.. jesus@eup.udl.es
[2] Mechanical Engineering Department. Universita Politècnica de Catalunya.
[3] Computing Department, Universitat de Lleida. xsirera@pie.xtec.es

Abstract

We present a work aimed to integration of multiagents systems in industry. There is an acute need for computer-aided tools to provide full support for mould engineering and to improve the competitiveness of mould manufacturers. The objective of the project is to develop a ***multi-agent system*** *which provides flexible integration of software tools belonging to the developers to be adapted to support the specific needs of the mould industry, providing users with a sufficient knowledge base to resolve problems and assist decision-making in the process of mould design.*

KEYWORDS: *blackboard, multi-agent systems, visual programming, mould engineering*

1 INTRODUCTION

The mould industry is under considerable pressure to reduce delivery times, improve quality and at the same time reduce costs. The mould industry is an industry with complex, unique products, highly dependent on expensive machinery and highly skilled, experienced and therefore costly personnel. Market demand is presenting this industry with a great challenge.
One of the possible ways to overcome this challenge is to support this industry with computer aided technologies.
Mould engineering and manufacture still depend heavily on craftsmen, basically on operatives with considerable experience and skill, used to working with traditional systems, not familiar with modern computer-aided technologies and reluctant to work with them.
The computer-aided systems currently on the market are useful only for some aspects of the mould engineering process. Furthermore, all of them adopt a general approach and do not provide sufficient resources for resolving the specific problems of mould engineering.
Each system, moreover, has its own complex user interface and is not well integrated with other systems.
For this reason, European mould-manufacturing companies have a considerable need for an easily usable integrated system with a single user interface, permitting management of the mould project, together with calculation and design, in order to provide a better design and better manufacturing results.

2 THE MULTI-MOULD PROJECT

The objective of the project is to develop a multi-agent system which provides flexible integration of software tools belonging to the developers to be adapted to support the specific needs of the mould-maker for the mould project, supervision, design and calculation of the mould.
The system will have a single user interface adapted to the user's language and culture, independently of the software packages used, making it easy to access and use in a transparent manner.
The system will provide users with a sufficient knowledge base to resolve problems and assist decision-making in the process of mould calculation and design, while supporting processing of users' specific know-how.

3 MULTI-AGENT SYSTEMS

In the previous paragraphs we have presented the type of problems we need to solve in the mould industry using computer systems, heterogeneous software products and experts in the domain and in the packages.
We need a problem-solving model for organising reasoning steps to construct a solution to the problem and co-ordinate the different actors of the problem (packages and experts). An architecture that fits with the characteristics of these kinds of problem is the blackboard architecture that was described back in 1962.[Newell, 1962]

The blackboard problem solving metaphor is very easy. It includes several agents gathered around the blackboard, looking at the pieces of information that are exposed, reading, thinking and writing conclusions
The blackboard architecture fits easily as a problem-solving model and as a framework for the implementation of architecture to solve the types of problems that we present here.
We will integrate the experts and the packages in the systems in the form of agents using the blackboard as the medium to define the problems and to monitor their solution.
In order to progress in the development of the architecture we have develop a type of specialised agents based on weak definition of the agents defined by Woolridge. [Woolridge and Jennings, 1995]
Our interest in agents based on definition is to see agents as concurrently executed software process that encapsulates some state and is able to communicate with other agents via message passing.
This weak notion of agent is the object of an emerging discipline, agent based software engineering:

Agents communicate with their peers by exchanging messages in an expressive agent communication language. While agents can be made as simple as subroutines, typically they are larger entities with some sort of persistent control [Genesereth and Ketchpel, 1994]

Agents in our architecture are processes that communicate with the blackboard using a communication protocol and language to solve a specific part of the problem. Agents can communicate with other agents using the blackboard as a co-ordination mechanism.

3.1.1 The state model as the problem solving model

The state model is the problem-solving model used in the blackboard to represent the problem. A state is a set of conditions or values that describe a system at a specified point during processing. The state space is the set of all possible states the system could be in during the problem solving process. State space representation solves problems by moving from an initial state in the space to another state, and eventually to a goal state. The movement from state to state is achieved by the means of operators. A goal is a description of an intended state that has not been achieved. The problem of solving a problem involves finding a sequence of operators that represent a solution path from an initial state to the goal state.
State space techniques have been used extensively to model problems in engineering applications [Brooks, 1983], management decision problems [Marshall et al, 1987] and in the implementation of expert systems [Badiru, 1992].

3.1.2 Visual programming and monitoring

A very important question to have in mind is that the type of experts that collaborate in the type of complex problems are not in general experts in computing, so it is important to provide them with a tool to allow them the design of the problem in the blackboard easily.
The blackboard metaphor is represented by green client windows with a toolbox and the mouse is the chalk to draw the problem-
The problem then could be drawn using a visual idiom [Cooper, 1995], ellipses, arrows and icons to represent states, transitions and agents.

4 APPLICATION OF MULTI-AGENT ARCHITECTURE TO THE MOULD PROJECT

We must think of mould design as a really complex problem.
The objective is to develop a system capable of realising the complete design of a mould for plastic pieces.
Therefore, it will be necessary to develop CAD applications for the display of projects in 2D and 3D, as well as to develop FEM systems to study thermal and mechanical behaviour of the mould; at the same time it will be necessary to create specialised systems to include the knowledge of expert Mould designers as well as several different modules which will be incorporated in a flexible and useful architecture.
It's easy to understand that all the actions performed by the different applications are very varied and complex and they imply collaboration between all the elements of the system.
For this reason, the use of a multiagent architecture is required in order to allow the system to control a series of such heterogeneous applications.

5 ARMAG architecture

We must think in ARMAG like a distributed multi-agent architecture based in a concurrent blackboard model [Engelmore, 1988]. This architecture is oriented to solve complex problems and uses state space model [Newell, 1962] like a paradigm of its resolution.
ARMAG architecture is composed of two different modules, the **Blackboard Agent** and the **other agents**, which collaborate in the solution of problems.

5.1 Blackboard Agent

This agent is the kernel of the architecture. It consists of two main modules included in a single workspace. One module, the *Blackboard*, controls and plans the problem while the other module is dedicated to agent registration and communication between Blackboard and the other agents. *CRM* (Control and Register Module);
On the one hand, the Blackboard consists of a *Visual Editor* with all the tools required to define the logic of the problem. On the other hand, it consists of an *Execution Engine* capable of solving the problem.

5.1.1 Blackboard Visual Editor

A very important feature in the Blackboard is its visual programming environment, which allows the user to design the problem in a visual way without requiring a high computing knowledge. The Visual Editor follows the visual metaphor of a blackboard where we "draw" the state diagram which defines the problem. During the design of the strategy, the user registers the different agents and establishes communication features.

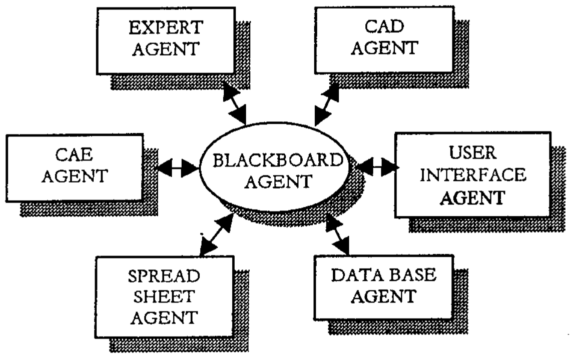
5.1.2 Execution Engine.

It's the module, which executes the problem defined by strategy in the Visual Editor.
Distributed execution. It provides the execution of problems in network distributed agents (Windows and Unix) taking advantage of the performance offered by current workstations. Agents, both local and remote, are launched automatically and simultaneously by the Execution Engine. Communications between these and the Blackboard are established by sockets.
Concurrent Execution. The Execution Engine allows the execution of actions in a concurrent way, reducing enormously the time to solve the problem. That is to say, we can design the problem by making the different actions run simultaneously. The Engine avoids the execution of two or more actions in one agent at the same time, if the agent doesn't support this feature.

Stopping execution. During the problem's execution, the Execution Engine allows the user to stop the process, as well as saving the resolution status and restarting the execution of the problem at the same point where it was left.
Problem's resolution status. Another important task to be performed by the Execution Engine is to communicate to the user the problem's resolution status, displaying information of the process at every moment: running actions, data modification, communication between agents, etc.
This way, we allow the user to have an exhaustive monitoring of the problem's solution.

5.2 Other agents.

We are refering to the rest of the agents that colaborate to solve the problem and which are specific for each



problem. The first thing to consider is that those called agents are, in principle, applications designed to perform specific actions. Therefore, an important task to do is to find the way to convert these applications into agents.That is to say, we must design an interface, which allows applications to interact with the Blackboard.
In conclusion, we can see the architecture of the agents as an application and an interface, which connects the application with the Blackboard.
In fact this is a high level overview of the architecture of agents. The real architecture of agents consists of four levels [Lorés 1994].

User interface agent. It performs all users' interactions, keeping a constant dialog with the user during the entire process of solving the problem.
CAD agent. It's based in a CAD application, which performs the graphic design of the Mould: situation of the relative components, size components, etc.
Expert agent. It's a knowledge-based system called MILORDII. This is for problems that in order to be solved need human experience.
Data Base agent. It performs data base management through ODBC.
This agent includes different kinds of databases:

- Plastic data base.
- Standard Mould components database.
- Injection moulding machines database

Spreadsheet Agent. This agent calculates any operations required during the problem solution process.

CAE agent. (FEM). It's a Fortran application dedicated to finite element calculation. Interaction with this agent is solved using BEM(Back End Manager) [Prat el al, 1990]. BEM is designed to solve troubles with numerical packages management. It performs all these operations to check the correct performance of the Mould

6 CONCLUSIONS

Application of the MAS in the Computing System which is being developed permits integration of the various *agents* which take part in the process of design and calculation of the moulds and also permits unification and simplification of the user interface, adapting it to user needs and potential.
The user-designer of the System will find integrated in it all the tools necessary for design and calculation of moulds through a single interface, and as well as having a whole range of knowledge from various experts, can himself extend these with own know-how and customise them for own needs, while easily editing and saving own strategies. Furthermore, users have all the necessary data bases on plastic materials and metallic materials for injection moulds, standardised modular mould elements, injection machines, etc., which can be added to over time.
All this potential which the System offers to users will permit considerable reductions of mould calculation and design times, thereby increasing the company's chance of being competitive in the face of others which do not have such a system.

REFERENCES

Badiru, Adedeji B (1992). Expert systems applications in engineering and manufacturing. Prentice Hall International.
Brooks, Rodney A., «Solving the find-path problem by good representation of free space». IEEE Transactions on systems, man and cybernetics. Vol SCM-13, N° 3, March/April 1983, pp 190-197
Cooper, Alan (1995). About face. The essentials of user interface design. IDG Books.
Engelmore, Robert (1988). Blackboard systems. Addison and Wesley
Genesereth, M. R. And Ketchpel, S.P. (1994). Software agents. Communications of the ACM, 37(7):48-53
Jennings N. R.et al (1966). Using ARCHON to develop real-world DAI applications for electricity transportation management and particle acelerator control. IEEE expert.
Lorés, Jesús (1994). «Disseny i desenvolupament d'una arquitectura distribuïda multiagent per a resoldre problemes complexos». PhD..
Marshall, G; T,j. Barber and J.Y. Boardman. «Methodology for modelling a project management control environment». IEEE proceeding-D. Control theory and applications. Vol 134, Part D. N4. July 1987, pp 278-285.
Newell, A. (1962). Some problems of the basic organization in problem-solving programs.In: Proceedings of the second conference on Selg Organizing systems. Spartan Books.
Woolridge, M.J.; Jennings, N. R. (1995). Intelligent agents. Theory and practice. The knowledge engineering review 10(2) 115-152.

AUTOMATION OF DEVELOPMENT OF SEQUENTIAL DIAGNOSTICS SYSTEMS IN MEDICINE AND ENGINEERING WITH THE HELP OF THE MULTI-AGENT SYSTEM "ARROW"

Boris. V. Maryanchik[1]

[1]*Dr. N., Moscow Research Institute of Pediatrics and Child's Surgery, Taldomskaya 2, 127412 Moscow RUSSIA,* Sasha@Gdev.msk.ru *, tel.(095)163-78-82*

Abstract

This work covers development of virtual networks method and soft computing on Bayesian logic for the automation of the knowledge base formation and finding out the most important additional information for running the sequential diagnostics. It is stated there that the increase in efficiency of the knowledge base development from 5-10 up to 50-100 diagnoses a month can be achieved owing to the multi-agent technology. After knowledge base is cut in work the application is ready and it can be used without carrying out the tests and refinements, if a data bases are not corrected.

Keywords: *multi-agent systems distributed intelligence, knowledge base, interdisciplinary investigation, recommendation automation.*

1. Introduction

The necessity for sequential diagnostics arises in any practically important case when the precheck information partially is not correct or is insufficient for recognition of illness of the person or defect of the device. The sequential diagnostics systems (SDS) give the recommendations about the most important additional information, which needs to be received in the appropriate analyses or procedures.

The user can apply the new tactics of diagnostics using one of the SDS [1,2,3]. As a rule, when a physician assigns investigations he tries to reduce uncertainty of one diagnosis in a differential row[2] to support or to reject it. The work of SDS is directed to a maximum reduction of the uncertainty of the whole row of the diagnoses. It allows to give more exact recommendations to optimize the process of diagnostics in cost and time.

Keith [4] has shown, in his work, those recommendations of a computer system based on the probability approach should be more exact, than those of an expert's as the system can take into account the greater number of factors. The problem lies in impossibility, at least in medicine, to receive rather exact data [5]. The solution of the problem can be obtained with the help of the virtual network method and soft computing on Bayesian logic [3] which are based both on knowledge and data.

The purpose of the work is to show employment of the method and computing for SDS development with the help of the multi-agent system "Arrow".

2. Transformation of a Data Base with the help of virtual networks in a Knowledge Base.

In applying this method in_medicine the data base includes the lists of: a) of the diagnoses representing full incompatible group of events on condition of manifestation in a patient the characteristic symptom-complex, б) the basic and additional signs sufficient, but not obligatorily necessary for statement of the diagnosis, with their (following a terminology [6] ") interval probability uncertainty " manifestations in the appropriate diagnoses._If the expert includes the additional signs into the basic ones, the program itself determines his mistake. Shown in

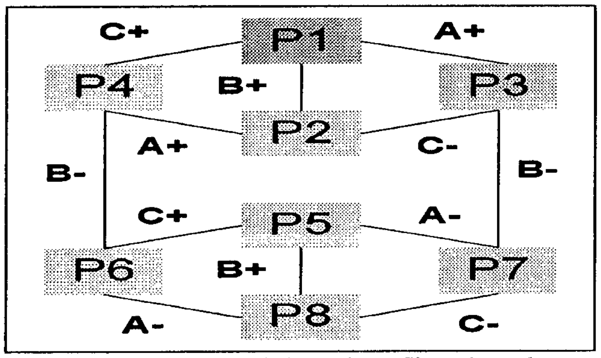


Fig.1 is the virtual network for 3 signs. Sings dependence, if it has place, is also taken into account by the program. A +, B +, C + - are signs present in illness manifestation,

Fig. 1 virtual networks for 3 Signs

A -, B -, C- are signs absent in illness manifestations, P1-P8- expert's estimations of probabilities of illness manifestation. Virtual networks similar to neural use a regression nonlinear model. In contrast to neural, the model here is the Bayes formula, knowledge discovery is reduced to a solution of the inverse problem: on estimations of probability the likelihood statistics Ls and Ln [7] are calculated, in this case the interval probability uncertainty is used as a restriction. The method works at uncertainty increase up to a maximum, but its accuracy decreases.

[2] The list of possible diagnoses one of which is supposed to be true is called a differential row of diagnoses

Other restriction following from the Bayes theorem is the requirement to eliminate the conflicts, when an expert implicitly attributes different weights to the same signs in different manifestations of the same illness while making the probability estimations.
The calculated values Ls and Ln are included in a knowledge base. They can be interpreted as weights of signs, describing a force in supporting or refutating the illness diagnosis, they can be checked in experiment and used for programming the recommendations in any field where the Bayes theorem is applicable.

3. Cooperative work of the agents in forming a knowledge base and supporting the sequential diagnostics

3.1 Cooperative work of the agents in forming a knowledge base

In Fig.2 shown is the work of Example's designer agent (E) and knowledge base agent (K) in forming the knowledge base. E receives from data base (DB) an information about the signs and their interval probability uncertainty, forms the network examples (see Fig.1 and Tab. 2) as combinations of appropriate signs, receives from K the boundaries_ (calculated depending on the uncertainty) in which an expert estimations can be found. User (U) (in this case an expert) informs K about the estimations of illness probability on each sample, K checks estimations for correspondence to restrictions,

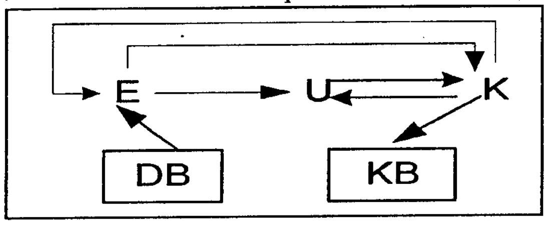


Fig. 2 Cooperative work of the agents in forming a knowledge base

indicated in item 2. If the estimations do not satisfy the check, K returns them to U and shows those, which should be revised if the estimations are satisfactory, the likelihood statistics (see item 3.1.1) are defined and knowledge base (KB) is formed.

Table 1

Fragment of estimations of manifestations of Tuberous sclerosis by the experts

Signs	Samples		
Depigmented spots	-	+	-
Brain calcificates	-	+	+
Internal organs	+	-	-
Cortical tubers	-	-	-
Renal polycystosis	-	-	-
Min	0	1	0.02
Max	0.96	1	0.99
Expert Conclusions	0.1	1	0.2

"+, -" – Availability or lacks of sign it a hypothetical patient.

3.1.1 Knowledge base agent forms a virtual network (VN) shown in Fig. 1. VN is used to solve the variational task for a spherical or cubic norm [8] (1) with optional restrictions (2)::

$$\|\vec{P}_e - \vec{P}_b(\vec{L}_{sn}, X)\| = \min, \quad (1)$$
$$\vec{L}_{sn} \in \Omega, \quad (2)$$

where

$\vec{L}_{sn} \in R^{2n}$, $L_{sn} = \{Ls_i\} + \{Ln_j\}$
$1 \leq i \leq n$, $j = 1 + n$,
n- number of basic signs, i- N_0 of a sign, $\vec{P}_e$ - vector of an expert estimations, $\vec{P}_e \in R^k$, $\vec{P}_b$ -- nonlinear 2n power operator, transforming by the Bayes formula the likelihood statistics in probability of hypothesis, k - number of network examples, $2n \leq k \geq 2^n$, X- matrix of parameters containing kn binary values of signs of network examples, Ω - area of possible values of likelihood statistics. In a common case the task (1,2) is incorrect: the insignificant changes in search of parameters and entry conditions lead to the large changes in the sought for vector [9,10]. The causes are nonlinearity of the variational equation (1) and multi-dimensionality of a vector space R^{2n}. The topology of the hypersurface under check is characterized for such functionals with narrow fancifully oriented hollows and with many local minimums. The standard methods of searching for global minimum of such functional do not exist. The approximate solution can be obtained with the help of the author's program, on the base of algorithms of casual and regular searching one by one.
After the likelihood statistics are defined, the knowledge base agent eliminates out of number of the basic signs those with values of a statistics Ls≈1 and Ln ≈ 1.
The forming efficiency of the knowledge base with the help of the example's designer agent and knowledge base agent depends on the number of basic illness signs and in the range from 8 up to 3 signs makes 50-100 diagnoses a month.

3.2 Cooperative work of the agents to support the sequential diagnostics

Presented in Fig.3 is the cooperative work of the Scenario Manager agent (S), Advice-giver agent (A) and Explainer agent (Ex) to support the sequential diagnostics.

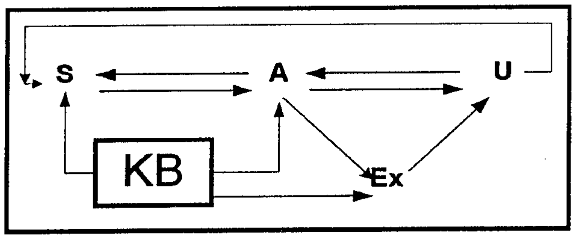


Fig.3 Cooperative work of the agents to support the sequential diagnostics

S receives from U the list with N signs of a patient and, using KB, tries to construct of them a differential row of the diagnoses. If the row is empty, it informs U that an input error of one or more signs has taken place and, deciding the combinatorial task, creates k combinations from N in N-1, N-2...signs for k differential row of the diagnoses, ranges them with the help of A. The each row

S ranges diagnoses by probability, and all rows by maximum probability of the diagnosis, found in it. Then S offers U to investigate them in the specified or random order. (S gives the developed in system "Arrow" applications stability to the signs input errors and allows to make some diagnoses in one patient, if they occur in different differential rows. In reviewing of the selected by U row, A with the help of algorithm described in item 3.2.1, recommends U the most important additional analyses and procedures. If U needs the explanation of the recommendations, they are formed with the agent E on the ground of information obtained from A and from the knowledge base.

3.2.1 Advice-giver agent with the help of computing made on the author's algorithm (3) gives the recommendations about realization of the most important additional analyses and procedures. Used in this computing are likelihood statistics from the knowledge base, processed are the probable intervals of uncertainty for the diagnoses, taken into account an actual force of manifestation of signs and ensured the solution stability against errors.

According to definition of Zadeh [11] such computing can be described as soft computing though it is fulfilled with the use of a probability measure of uncertainty, but not a fuzzy measure.

Mainly responsible for the most important additional analyses or procedures are the signs, availability or lack of which in a patient can supply minimum or near to minimum value of goal function μ:

$$\mu\left(P_{\max} P_{\min} L_{sn} \left\{V^0_{ij}\right\}\right)_{V_{sl}=1\vee 0} \approx \min \quad (3)$$

$$P_{\max} = P_{\max}\left(\left\{V_{ij}\right\}|_{V_{ij}\in\Omega_1}, V_{sl}\right)$$

$$P_{\min} = P_{\min}\left(\left\{V_{ij}\right\}|_{V_{ij}\in\Omega_2}, V_{sl}\right)$$

Here μ - current uncertainty of investigated differential rows, P_{max}, P_{min}- vectors max and min diagnoses probability, which form the probable intervals of uncertainty, i- number of an unknown sign in illness J, $i \neq s$, j=1,2...r, r- number of the diagnoses in a differential row, V_{ij} – possible value of still not investigated sign, V^0_{ij} - actual value of investigated sign, V_{sl} - tested sign values (1 or 0), Ω_1: $V_{ij} \equiv 1$, Ω_2: $V_{ij} \equiv 0$, L_{sn} - matrix of likelihood statistics.

4. Practical outcomes

The author's programs on realization of methods described in this paper were used for the development of system "West-Syndrome", which was included in international catalog "Epilepsy in focus".

5. Conclusion

In this work described is the method of automation of knowledge discovery with the use of the implicit conflicts detection procedure in an expert with Bayesian logic, as well as the method of automation of recommendations programming. In the opinion of the author, it allows to simplify the work both of the experts (as the faultlessness of their conclusions becomes less important), and the developers (as a smaller volume knowledge in the field of the artificial intelligence is required from them) and to considerably reduce the cost of SDS development.

The efficiency of the knowledge base development is 50-100 diagnoses a month, it can be achieved owing to the multi-agent technology. After knowledge base is cut in work the application is ready and it can be used without carrying out the tests and refinements, if a data bases are not corrected.

Gratitude

The author is grateful to the experts in the field of medicine prof. B.A. Kobrinsky, prof. P.A. Temin, d.m. V.V. Dlin and other researchers of the epileptology and nephrology departments of Moscow Research Pediatrics and Children's Surgery Institute, who used and helped in the improvement the described methods for the development of some SDS.

Bibliography

1. Management of uncertainty in medicime/Cohan Paul,..//6th Annu.Int.PhoenixConf.Computandcommon,Scottsdale,Ariz. Feb.,25-27,1987 Conf. Proc.-Washington (D.C.).,1987.-C501-506
2. J.B. Parris, A. Vencovska. On the applicability of maximum entropy to inexact reasoning.// Int. J. Of Approximated reasoning, 3,1,1989
3. B.V. Maryanchik, About Application of Virtual Nnetworks and Soft Computing in Medical Diagnostics// Sixth national conference with international participations, CAI'98, 5-11 October, Pushchino, Russia, 1998 , v.1 p319-324
4. Expert system should be more accurate then human experts: estimation procedures from human judgement and decisionmaking/Levi Keith//IEEE Trans.Syst., Man, and Cybern.-1989.-19,N3.-C.647
5. Shortliffe E. Computer Based Medical Consultetions MYCIN. American Elsevier, New York, 1976
6. Gorodetski V.I. Consistency of Knowledge Bases with Quantitative Uncertainty Measures// Sixth national conference with international participations, CAI'98, 5-11 October, Pushchino, Russia, 1998, v.1 p.100-107
7. Forsyth R(ed) Expert Systems. Principles an Case Studies, Chapman and Hall, London, 1984]
8. V.A. Trenogin, The functional analysis. –Moscow, Science, 1980
9. Maryanchik B.V., The Method virtual statistics and its use in partner systems for computer diagnostics // The Computer chronicle, Moscow, 1996. N5. ,c. 65-74
10. Kobrincky B.A., Maryanchik B.V.and other, Application of a process engineering of virtual statisticians for development of a medical diagnostic system based on knowledges. // The Computer chronicle. 1997. N4 , c.3-12.
11. Zadeh L.A/ Fuzzy logic, neural networks and soft computing // Communications of the ACM.-1994.-Vol37, No 3.-P.77-84

A MULTI-AGENT SYSTEM FOR SUPPORTING JOB-SHOP SCHEDULING[1]

Zhongzhi Shi Yunfeng Li Wenpin Jiao Hu Cao

Institute of Computing Technology, Chinese Academy of Sciences
P.O. Box 2704-28, Beijing 100080, P. R. China
E-mail: {shizz@envst-1.ict.ac.cn}

Abstract
We present a multi-agent system for supporting job-shop scheduling which is named MASJSS. In this paper, the architecture of MASJSS is given, as well as the multi-agent decision-making process. Based on a community of agents, MASJSS is able to decide whether to accept an order and how to accomplish the accepted order. For scheduling process, an incrementally cyclic procedure is proposed which involves three stages, (1) goal-driven partial decision making; (2) conflict recognizing; (3) conflict resolving by using constraint heuristic backjumping.

Key Words: *multi-agent system, negotiation, constraint reasoning*

1. Introduction

Multi-agent system offers a new technology for manufacturing enterprise integration, and a lot of agent based systems have been developed to support various dimensions in manufacturing environment[1][2]. The so-called job-shop scheduling problem, which decides whether to accept an order and manufacture planning, is important for an enterprise to respond quickly to the ever-changing marketplace needs. The scheduling problem has some particularly properties, for instance, (1) time is an important factor in scheduling, (2) resource allocation problem exists in decision process, and (3) interdependence among agents exists.

Based on the above observations, we design and implement a multi-agent system, called MASJSS, to support job-shop problem solving. In MASJSS, agents are employed to simulate manufacture departments of a factory, the community of agents are used to simulate the running of the factory, and the job-shop scheduling problem will be solved by these agents via a decision making process.

2. Architecture of MASJSS

The architecture of MASJSS system can be outlined as three layers (see Figure 1). The first layer is intended to provide intelligent decision support techniques for manufacture department agents to make partial decisions. The second layer is the interface between manufacture department agents and decision support components. The third layer is the kernel part of MASJSS, in which agents represent manufacture departments respectively and can use all kinds of decision support techniques under the coordinating of the facilitator agent

3. Decision Making Process

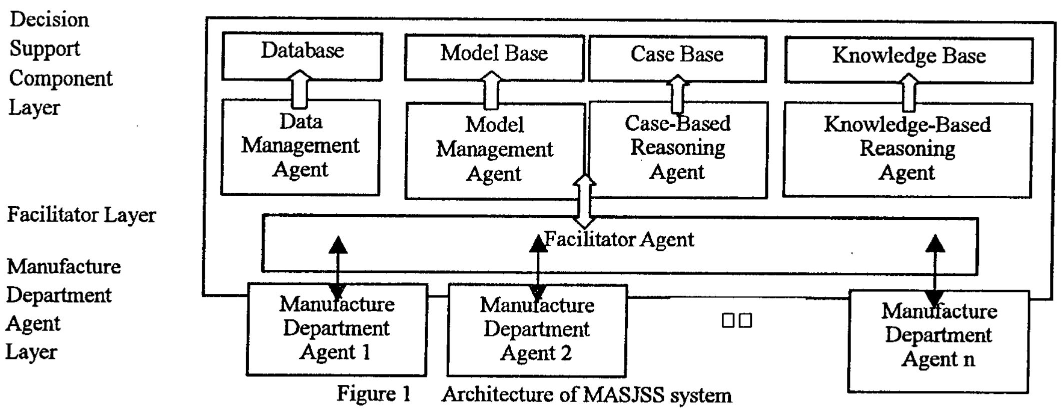

Decision Support Component Layer
Facilitator Layer
Manufacture Department Agent Layer
Database
Model Base
Case Base
Knowledge Base
Data Management Agent
Model Management Agent
Case-Based Reasoning Agent
Knowledge-Based Reasoning Agent
Facilitator Agent
Manufacture Department Agent 1
Manufacture Department Agent 2
Manufacture Department Agent n

Figure 1 Architecture of MASJSS system

There are two tasks for job-shop scheduling: (1) to decide whether to accept an order, and (2) how to accomplish the order. MASJSS focuses on the first task, if the order is accepted, the manufacturing plan will be given simultaneously as the «byproduct» of scheduling process.

In job-shop scheduling problem, agents are

[1] Supported by National 863 Hi-Tech Program

interdependent, and the negotiation process is interwoven with partial decision making processes. We view the job-shop scheduling process as an incrementally cyclic procedure composed of three steps: (1) Goal-driven partial decision making; (2) conflict recognizing; and (3) conflict resolving.

3. 1 Goal-Driven Partial Decision Making

In job-shop scheduling problem, the order of agents to make their partial decisions is conversely to positions that they are in a working procedure, which will be continued until all relevant agents make their own decisions (if the order is acceptable). We call such a multi-agent decision making process as goal-driven decision making.

According to decision making orders, we group agents into agent sets which are denoted as AS_1, AS_2□□□□AS_m. The decision making process can be represented as algorithm GDDMP (Goal-Driven Decision Making Process):

Algorithm GDDMP:

```
FOR t = 1 TO m DO
 BEGIN
  For all agents in AS[t]:
   BEGIN
    Agents make their partial decisions respectively;
    Send their results to global blackboard;
    Conflict recognition;
    IF there are conflicts exist, do conflict resolution;
   END
 END
```

3.2. Conflict Recognizing

There are two types of conflicts in job-shop scheduling problem: (1) time constraint conflict, which means that agents can not finish the order in time; (2) resource constraint conflict, which means agents' needs for some particular resources exceed the limitations.

For each agent set AS_1, AS_2, □□, AS_k, the maximum demand times are T_1, T_2,□□, T_k, and the total possible time is TD. If T_1+T_2+□□□T_k > TD, which means that the order can not be finished in time, time constraint conflict exists and negotiation process should be started.

Resources in this paper refer to the economical resources necessary for manufacturing any products. Resource constraint conflict is recognized by using *resource-agent demand matrix*, whose rows represent all kinds of economical resources E_1, E_2, □□E_m, and columns represent agents of MASJSS system A_1, A_2, □□A_n. The resource-agent demand matrix is defined as $M_{m,n}$, element $m_{ij} = 0$ if E_i is not necessary for Aj, otherwise m_{ij} equals to the quantities that A_j required. After partial decisions being made, the quantities of demand for resources will be filled in this matrix. If the summation of one matrix row is greater than the limitations, the resource constraint conflict exists.

3.3. Conflict Resolving by using Heuristic Backtracking

Once conflict exists, constraint relaxation and propagation techniques will be used. The assignment order of constraints is determined by agent involved degree and resource contention degree that defined as follows.

Definition 1. For a particular agent α, **Agent Involved Degree** represents how many kinds of resources that used by α to archive its goal. It equals to the number of non-zero elements in the column where α is located.

Definition 2: For a particular kind of economical resource *e*, **Resource Contention Degree** represents how many agents require this kind of resources. It equals to the number of non-zero elements in the row where *e* is located.

After an agent α has made its partial decision, if conflicts are detected, it first attempts to propose other proposals. Once there are no other alternatives, constraint heuristic backjumping is used to resolve conflicts. The central idea of constraint heuristic backjumping is when an agent failed to make a non-conflict decision, the algorithm will jump back to the agent which has the tightest relations. The conflict resolution process is represented as algorithm CRHBJ (Constraint Resolution by Heuristic BackJumping):

Algorithm CRHBJ:

```
WHILE conflicts exist DO
 BEGIN
  Select an agent A to jump back;
  IF A = NULL THEN RETURN with failure;
   A re-instantiates involved constraints;
   IF A successfully re-instantiate THEN
     BEGIN
      Constraint propagation;
      Conflict Recognition;
    END
 END
```

4. Related Works

Sycara et.al studied distributed constraint heuristic search (DCHS) and used job-shop scheduling problem as its experimental domain [3]. She selected activity as decision unit, while our approach select the whole manufacture department to make goal-driven partial decisions. Time in DCHS is the resource aggregation dimension while it is decision dimension in our approach. Conry et.al presented multistage negotiation protocol that structured negotiation process in three phases[4]. The essential difference between our approach and hers is that we treat partial decision making in an incremental manner, and negotiation process interweaves with the partial decision making process.

Future researches will concentrate on expand the goal-driven decision making process to more applicable domains and to propose a complete multi-agent negotiation model.

References

[1]. Timothy J. Norman, Nick R. Jennings, Peyman Faratin, E. H.Mamdam, Designing and implementing a multi-agent architecture for business process management, Intelligent agents □, agent theories, architectures and languages, J. P. Muller et.al eds, pp261-275

[2]. J. C. Beck, Mark S. Fox, Supply chain coordination via mediated constraint relaxation, in Proceedings of the first Canadian workshop on distributed artificial intelligence

[3]. K. Sycara, S. Roth, N. Sadeh, M. Fox, Distributed

constraint heuristic search, IEEE transactions on system, man and cybernetics, Vol.21, No.6, pp1446-1461, 1991

[4]. Susan E. Conry, Kazuhiro Kuwabara, Victor R. Lesser, Robert A. Meyer, Multistage negotiation for distributed constraint satisfaction, IEEE transactions on system, man and cybernetics, Vol.21, No.6, pp1462-1477, 1991

A COGNITIVE AGENT FOR SOCCER GAME

Leo A. Stankevich

Saint-Petersburg State Technical University, Politekhnicheskay 29, phone (812) 247-42-14, e-mail: stankevich@phtf.stu.neva.ru

Abstract

Soccer (association football of robot) makes a good example of the problem of the real world, which is moderately abstracted. This game has chosen as one of standard problems for study on multi-agent systems. We are developing the soccer agent, based on the cognitive approach. Our soccer agent can learn whether he shoots a ball or pass it because the agent has the neurological modules inside..

Keywords: *multi-agent systems, soccer agent, neurological module, client-server programming.*

1. Introduction

The distributed artificial intelligence (DAI) and multi-agent systems (MAS) research directions became activity at present. We see wide use of MAS-technology application for design of DAI real-time control systems of the robots grouped with the common work goal. The complexity of the design DAI real-time systems with MAS-technology has caused the machine learning (ML) use. The computation complexity is being replaced with system learning. The max reaction to situation can be gotten only taking dynamic changes into account.

Soccer (association football of robots) is a team game in which the players have a cooperation [1,3]. This is a real-time game where situation changes dynamically. Soccer was chosen as one of problems for studying on multi-agent systems [5,6]. We are designing the soccer agent using the cognitive approach. We presented an algorithm of the decision making for a cooperative action among soccer players. We developed the learning modules for partial implementation of decision making at high and low behavior levels of soccer agent. This soccer agent can be used as a client of Noda's Soccer server for participation in simulation league of RoboCup [4,7]. In the Saint-Petersburg State Technical University has organized soccer team for participation in RoboCup [2].

The following specific features of our agent are considered in this paper.

1. The agent is designed as a cognitive system that is the learning intelligence system with nervous system behavior, function, and structure. Agent's knowledge is produced by learning in a work process. The knowledge is kept and used in the associative neurological form.
2. The specific evaluation function, described below, used for a decision making. This function is used at high level control with a decision tree. It can be tuned during the learning process for adaptation to environment..
3. The middle level behavior function set is used for agent's individual tactics with operative change for adaptation to game situations.
4. The specific learning module for automatic interaction among others agents during game. It improves coordination at attack or defense.

2. Agent's behavior

The soccer agent has a multi-level behavior like football man does. The agent is based on simulating several behavior functions that formally can be mapped as:

$$SBF = (CF, MF);$$
$$CF = insf(EF, DM, IA, BL);$$
$$EF = insf(TM, CD); DM = insf(AT, DF);$$
$$IA = insf(MS, RL, FM); BL = insf(CH, GM);$$
$$MF = insf(Pass, Dribble, Avoiding, Pressing, Intersect);$$
$$MF = compf(Kick, Turn, Dash, Inhibit),$$

where SBF - Soccer behavior functions; CF and MF - Cognitive and Motion functions; EF - Evaluation function; DM - Decision making function; IA - Interaction function of players; BL - Behavior learning function; TM - Teammate positions; CD - Contradictor positions; AT - Attack decision; DF - Defense decision; MS - Message change; RL - Role of agents; FM - Formation from roles; CH - Coaching learning; GM - Game learning. These functions determine the team strategy and tactics in the battle with contradictors. Low level behavior includes individual motion skills of players (MF). There are Pass, Dribble, ball Intersect, Avoiding of obstacles, Pressing on contradictors. The final actions of players include Kick, Turn, Dash, and Inhibit. The operations 'insf' and 'compf' mean 'insert functions' and 'component functions'.

3. Evaluation function

The correct evaluation of situation and position of the players is fulfilled with special evaluation function. This function evaluates the position where the player has ball with relation to teammates and contradictors. The function for the i-th player is:

$$K_{ij} = F\{C_{ij}, W_{ij}, \Sigma_{q=0}^{n}[\alpha_q, P_i(q,M(t))]\}, i=1,\dots,11,$$

where C_{ij} - interaction force coefficient of i-th and j-th players from coach; W_{ij} - adaptive coefficient for mapping of the game experience with similar contradictors; α_q - prediction coefficient on period t; $P_i(q,M(t))$ - position evaluation with current position M(t) teammates and contradictors; F - constrained function. This function P is implemented on neurological module with learning.

4. Decision-making

The choice of pass, dribble and shoot is executed on decision tree. There are branches of the tree that help to decision making for player with ball or without it, to attack or defense in the current game situation. The decision-making algorithm includes:

- calculation K_{ij} ($j = 1,\dots,11$) for i-th player with ball;
- choice of of S perspective candidates on max K_{ij};
- calculation K_{qj} ($q = 1,\dots,S$) with respect to q-th perspective;
- choice of the action on max K_{qj} (pass or dribble); the player without of ball makes decision in branches of decision tree basing on hard algorithm.

5. Neurological module

The partial evaluation function $P_i(q,M(t))$ is implemented with neurological module that can learn on examples (patterns). This is associative fuzzy-logical neural network with layered structure. Formality it can be as:

$$NM = \{X, H_x, St, W, H_y, Y\},$$

where X and Y - input and output vector variables; H_x and H_y - hidden variables for activator of the module; St and W - structure and connection weights for the activator of module. This module is executes any vector-vector mapping $X \rightarrow Y$ on many examples (patterns) with learning. It can be used for implementation of individual skills, too.

The neurological modules based on fuzzy logic are used in the agent. The follow procedures are used in design:

- Fuz - fuzzification calculating the membership degree μ^q_{xi} of variable x_i to q-th fuzzy granule this variable;
- Wagr - weight aggregation calculating the membership degree μ^q_y of variable y_j to fuzzy granule this variable;
- Dfuz - defuzzification calculating the value of output variable y_j;
- Wupd - weight update at learning of the neurological module by examples of map X to Y.

Let us consider on the example of two input variables x_1 and x_2 and one output variable y. Let every variable is decomposed on the three fuzzy granules and membership function have the triangular form. Then the procedures look as:

$$\text{Fuz: } \mu^q_x = ((x-l_q)c_{ql}^{-1} \Leftarrow l_q<x<m_q) \vee ((r_q-x)c_{qr}^{-1} \Leftarrow m_q<x<r_q) \vee (0 \Leftarrow \text{else});$$

$$c_{ql} = m_q-l_q;\ c_{qr} = r_q-m_q;$$

$$\text{Wagr: } \mu^q_y = \vee_{i=1}^{i=9}(w^q_i \wedge_{q,r=1}^{q,r=3}(\mu^q_{x1}, \mu^r_{x2}));$$

$$\text{Dfuz: } y^q = 0.5(y^q_n+y^q_r) = 0.5(\mu^q_y/c_{qn}+\mu^q_y/c_{qr});$$

$$y = \Sigma_{q=1}^{K}\mu^q_y y^q / \Sigma_{q=1}^{K}\mu^q_y;$$

$$\text{Wupd: } w_{p+1} = w_p\,(p/p+1)+w_p(\mu^*_y/\mu_y)/(p+1);$$

$$p=1,\dots,n_p,$$

where l_q, m_q, r_q - coordinates of left, middle, and right points of triangle base in membership function, $\wedge$ - fuzzy intersection (min), $\vee$ - fuzzy unify (max), w_i - weight of structure connection of the module's activator.

6. Individual skills

The ball action individual components of agent are implemented on hard algorithm or neurological module. We used combined method. The agent's action such as shoot, run, turn dash, inhibit are executed on hard algorithm. The action as pass, dribble, intercept, avoiding, pressing, corner-kick are more complex agent's skills. They were implemented on neurological module with ML methods for tuning of dribble with avoiding the obstacles. The goalkeeper-agent has some specific features. It could predict ball motion direction and intersect its. The ML methods for the tuning of skills are used to intersect the ball, too.

7. Interaction of agents

The multi-agent soccer system is a client-server program environment. The agents are divided into two teams that have cooperation and collaboration. A team's agents have their own roles and formation. Let

$$CA = \{ca_1,\dots, ca_m\};\quad R = \{r_1,\dots,r_m\},$$

where ca_i - i-th agent; r_i - i-th role. Formation F is components U_j from roles R as

$$F = \{R, (U_1,\dots, U_k\};\ U_j \subset R;\ U = \{r_{i1},\dots,r_{ij}\}\ (r_{ij} \neq r_{iq}).$$

A mapping $CA \rightarrow R$ is flexible, since it depends on time. The roles can be assigned to different homogeneous agents. We have used special neurological modules for agent interaction. Formation can affect the agent's external behaviors by specifying inter-role interaction. Since roles can be re-used among formations, their formation-specific interactions cannot be included in the role definitions. Instead of it, the interactions are a part of the formation specification.

8. Conclusion

As a result, the cognitive soccer agent is designed. It has flexible behavior structure with learning. This agent is being tested for participation in RoboCup. The testing games have shown small advantage of this agent in team tuning that can be produced on game's time. However, after learning while playing, the next game is expected to be more successful.

Bibliography

1. Matsubara H., Noda I., Hiraki K. Learning of Cooperative Actions in Multi-agent Systems: Case Study of Pass Play in Soccer. // AAAI-96, Spring Symp. On Adaptation, Co-evolution, and Learning in MAS, 1996.
2. Stankevich L.A. There are competed the Saint-Petersburg soccer-robots // Sport Marketing, no. 1, 1999 (Rus. language).
3. Michael K. Sanota. Real-time intelligent behavior in dynamic environments: Soccer-playing robots. Master's thesis, Univ. Of British Columbia, 1993.
4. I. Noda. Soccer server: a stimulator of RoboCup // AI Symposium'95, Japanese Society for AI, 1995.
5. M. Veloso, P. Stone, K. Hun, and S. Achin. The CMUunited-97 Small Robot Team // RoboCup-97: The First Robot World Cup Soccer Games and Conferences, Ed. H. Kitano, Springer Verlag, Berlin, 1998.
6. P. Stone, M. Veloso. A layered Aproach to Learning Client Behaviors in RoboCup Soccer Server // Applied Artificial Intelligence, 12, 1998.
7. Result of RoboCup-97. Accessible from http://www.RoboCup.org.

THE REFLEXIVE MODEL OF THE AGENT'S NORMALIZED BEHAVIOR

Tatiana A. Taran

KPI National Technical University of Ukraine, Peremogy Av. 37, 252056 Kyiv, UKRAINE,
taran@pma.ntu-kpi.kiev.ua

Abstract

In this paper we suggest to apply the main ideas of reflexive theory to modeling of the intelligent agent behavior in a situation of a choice between various alternatives. The choice of agent depends of some set of norms which can be given to regulate the agent behavior. Sets of alternatives are ordered in such a way as to form Boolean lattices. We build the logical model of agent with normalized behavior described by means of Many-valued Boolean logics.

Keywords: *intelligent agent, reflexive behavior, many-valued Boolean algebra.*

1. Introduction

Modeling of the agent behavior and decision-making under conditions .of choice is an important problem whose solution has been attempted in a variety of ways. In the early 1960's V. Lefebvre proposed an approach in which the reflexive property of the human being is formalized [1,2,3]. The reflexive models by V. Lefebvre allow us to describe and predict some properties of human choice in conflict situation. In [4] application of principles of the reflexive control was suggested for modeling of agent's behaviour in a situation of binary choice. In this paper we develop the main ideas of reflexive theory to modeling of the intelligent agent behavior in a situation of a choice between various alternatives. We consider the situation when the choice of agent depends on some set of norms that can be given to regulate the agent behavior. That means that we build the reflexive model of agent with normalized behavior.

2. The reflexive approach

In reflexive approach the agent is regarded as a computational schema [2]. The process by which the agent makes a decision is likened to a sequence of binary choices, each taken as an elementary act of decision-making. An elementary choice is enacted on a bipolar scale with two poles, positive and negative. Under pressure from the outside world to choose one of the poles, the agent is able to choose one of the poles at each moment of time.

We consider the schema of agent consisting of three levels [2]:

$$behavior = perception^{knowledge^{intention}}$$

Perception is the first level where the pressure of the outside world influences to the agent. The second level is the agent's *knowledge* concerning the pressure of the outside world, an opinion which does not necessarily coincide with reality. The agent has certain *intentions* toward the choice of one alternative from a set of alternative.

This reflexive structure is described by the function of three variables:

$$Y = f(x1,\ x2,\ x3), \qquad (1)$$

where *x1* describes the pressure of the external world toward the choice of one alternative, *x2* the agent's knowledge of the external world's pressure, and *x3* the agent's intention. This function *Y* characterizes the agent's readiness to choose of one alternative.

In Lefebvre's works [2] a binary logic of morality is constructed in which the agent is capable of choosing one of the poles of an ordered binary scale: *bad* or *good*. If we regard the scale of choice as a continuous linear scale defined on the interval [0, 1], Lefebvre's model is a probabilistic model of the agent's behavior under conditions of choice.

3. Normalized Behavior of the Agent

In this work we will examine the behavior of the agent under conditions of choice, when there exists a set of alternatives. These sets of alternatives are ordered in such a way as to form Boolean lattices. The agent possesses the ability to choose one of the alternatives. These alternatives are regarded as *norms* regulating the conduct of the agent, and the Boolean lattices are regarded as *systems of norms.* The agent's behavior under the influence of partially-ordered sets of norms will be called *normalized behavior*. In the present work Boolean systems are used as the basis for *a logic of norms* describing the normalized behavior of the agent.

The normalized behavior of the agent is described by function (1). The world dictates to the agent the necessity of choosing *x1*, the agent imagines to himself the world's pressure as norm *x2*, and at the same time he has the intention of choosing norm *x3*. The value of function *f(x1, x2, x3)*is the norm that the agent is ready to choose.

3.1. Many-Valued Boolean Scales

We will regard many-valued scales formed on the partially-ordered sets *L* and they form a Boolean lattice. We shall consider at first a simplest case: scales consisting of four elements: $L = \{0, C, S, 1\}$. These elements can form a partially-ordered set such that $0 \le C \le 1$, $0 \le S \le 1$, and *C* is not comparable with *S*. Such a four-element scale forms a Boolean lattice [5].

The rhombus pictured in Figure 1 is a Hasse diagram of the Boolean lattice L, whose points correspond to the elements of the set L. The order relation between two points on the rhombus is indicated from bottom to top. The values C and S are opposed to one another. In the lattice L these two values are complementary one to another.

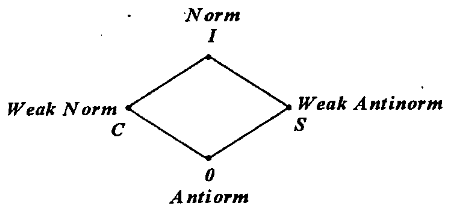


Figure 1: The Boolean scale L

Each point on the Boolean scale determines one value for the norm, that is, one of the alternatives for choice.
On the Boolean scale L we will call: the *supremum* of the ordered set L (the unit point 1), the *Norm*, the *infimum* of the ordered set L (the null point 0), the *Antinorm*.
The intermediary values lying between the *Norm* and the *Antinorm* will be called the *Weak Norm* and the *Weak Antinorm*. In general, if a certain value $x \in L$ is the *Weak Norm*, then the value $y \in L$ that is the complement of x in the lattice L is its *Weak Antinorm*.
Example.
In some cases a four-element scale can be formed out of a combination of norms and antinorms defined on binary scales. Imagine an agent in a situation of conflict. In this conflict *victory* is possible, but *defeat* is possible as well. *Victory* and *defeat* form a binary scale on which *victory* is the *Norm* and *defeat* the *Antinorm*. Also, *victory* can be achieved honestly, by following the rules, or dishonestly, by violating the rules. *Honesty* and *dishonesty* form another binary scale.
The two norms and antinorms can be combined: *dishonest victory* and *honest defeat, honest victory* and *dishonest defeat*. These four values form a partial order:
dishonest defeat $\leq$ *honest defeat* $\leq$ *honest victory,*
dishonest defeat $\leq$ *dishonest victory* $\leq$ *honest victory.*
Honest victory and *dishonest defeat* are now the Norm and Antinorm of a partially-ordered set of alternatives. *Dishonest victory* and *honest defeat* are the Weak Norm and the Weak Antinorm.
Now the choice occurs on a four-valued scale that is formed by the Cartesian product of two scales.
Formally, the Cartesian product of binary scales produces new combinations combining the norms and antinorms in various ways. In some cases, as in the example given above, the combinations of norms and antinorms can give rise to new *Norms*, *Antinorms*, and also *Weak Norms* and *Weak Antinorms*. It should be emphasized that the systems of norms are formed by the Cartesian product of the binary scales only in case its elements have an appropriate interpretation.

3.2. The Logic of Norms

On the Boolean lattice $L = \{0, 1, C, S\}$ it is possible to define a four-valued Boolean logic B_4 [5] in which the operations of disjunction ($\vee$) and conjunction ($\wedge$) correspond to the lattice operations of union and intersection: $x \vee y = sup(x,y)$, $x \wedge y = inf(x, y)$. The operation of negation ($\neg$) corresponds to the complement in the lattice L. The operation of implication is defined as $\neg x \vee y \equiv x \to y$. We shall call the four-valued Boolean logic constructed here *the logic of norms.*

4. Logical Model of the Agent

Let us consider an agent who finds himself faced with a choice among four alternatives corresponding to values on a Boolean scale of norms. We shall construct a logical model of the agent in which the values are defined on the set of logical values $L=\{0, 1, C, S\}$. The agent's readiness to choose one of the alternatives is described by a function in three variables [2]:

$$A1 = a1^{a2^{a3}} \quad . \tag{2}$$

An expression of the type a^b represents the logical implication: $b \to a$, defined in logic B_4. For this reason formula (2) is equivalent to the logical formula:

$$A1=(a3 \to a2) \to a1. \tag{3}$$

In the formulas (2) and (2) the variable $a1$ describes the pressure exerted by the outside world towards choice of one of the alternatives: *Norm, Antinorm, Weak Norm*, or *Weak Antinorm*. The agent's conception of the pressure of the outside world is described by the variable $a2$. The variable $a3$ describes the agent's intention of choosing one of the alternatives. The value of the function $A1$ describes the agent's readiness to choose one of the alternatives.

4.1. The Agent's Realistic Choice

The agent's choice is realistic if it coincides with his intention [2]. The solution of the equation

$$a1^{a2^{x}} = x \tag{4}$$

for all possible values of $a1$, $a2$ describes the agent's realistic choice.
In certain circumstances the agent possesses *freedom of choice*. The conditions under which freedom of choice is possible are defined by the values $a1$, $a2$, such that for any intention $a3$ this turns into the agent's readiness to choose. Then the following identity holds:

$$a1^{a2^{x}} \equiv x \tag{5}$$

Let us define the conditions under which an agent's capacity for realistic choice appears in our model, including free choice as a part of this notion. To do this we find all solutions for equation (4) for the 16 sets of values for the variables $a1$, $a2$. The solutions of these equations form three groups corresponding to determined choice and partially and completely free choice: *determined choice, partial freedom of choice, complete freedom of choice* (see table).

Determined choice	
${}_0 1^x = x$	$x = 0$
${}_1 0^x = x$	$x = 1$
${}_1 1^x = x$	$x = 1$
${}_1 C^x = x$	$x = 1$
${}_1 S^x = x$	$x = 1$
${}_C 1^x = x$	$x = C$
${}_S 1^x = x$	$x = S$
${}_C S^x = x$	$x = C$
${}_S C^x = x$	$x = S$

Partial freedom of choice	
${}_0 C^x = x$	$x = 0, S$
${}_0 S^x = x$	$x = 0, C$
${}_C 0^x = x$	$x = C, 1$
${}_C C^x = x$	$x = C, 1$
${}_S 0^x = x$	$x = S, 1$
${}_S S^x = x$	$x = S, 1$
Complete freedom of choice	
${}_0 0^x = x$	$x = 0, 1, C, S.$

Having surveyed all the solutions to equation (4), we see that there is only one case where it turns into an identity. Thus we can conclude that the agent has free will only in case $a1 = 0$ and $a2 = 0$, that is, when the world inclines him to choose the *Antinorm* and he has a correct knowledge of that pressure.

The remaining values of the function (4) describe the behavior of the agent when his choice is not realistic.

4.2. Generalization of Normalized Behavior Model in Many-Valued Boolean Logic

A many-valued logic of norms can be constructed on any Boolean lattice 2^n, where *n* is the number of binary characteristics entering into the lattice.

Generalizing the analysis of a model of the agent behaviour, defined on many-valued Boolean scales $L = 2^n$, it is possible to prove the following theorems.

Theorem 1. *The agent makes a realistic choice in that and only that case, if for anyone of a1, a2, a3, defined on a Boolean lattice L, the condition is satisfied:*

$$a1 \le a3 \le \neg a2 \vee a1 .$$

From the theorem 1 the conditions of a determined choice, partial and full freedom of a choice follow.

Theorem 2. *The agent has full freedom of a choice in that and only that case, if*

$$a1 = 0 \text{ and } a2 = 0.$$

Then a3 accepts any values from a lattice L:

$$0 \le a3 \le 1.$$

Consequence of the theorem 2.

The values $a1 = 0$, $a2 = 0$, with which there exist full freedom of a choice, are unique.

The following theorem generalizes possibilities of partial freedom of a choice.

Theorem 3. *Partial freedom of a choice is possible in that and only that case, if*

$$a1 < \neg a2 \vee a1.$$

Then the agent is free to choose any of alternatives $a3$ which belongs to a closed interval

$[a1, \neg a2 \vee a1]$.

Consequence 1 of the theorem 3.

If pressure of the world $a1 = 0$, and the evaluation of this pressure by the agent is a2, the agent is free to choose any of alternatives a3, which belongs to the ideal $[0, \neg a2 \vee a1]$.

Consequence 2 of the theorem 3.

If the pressure of the world is a1 and evaluation of this pressure by the agent is $a2 = 0$, then the agent is free to choose any of alternatives a3, which belongs to a dual ideal $[a1, 1]$.

Consequence 3 of the theorem 3.

The amount k of alternatives of a choice with partial freedom of a choice on a many-valued Boolean lattice with 2^n elements is defined by an inequality

$$2 \le k \le 2^{n-1}$$

and is multiple of the whole degree two.

The following theorem defines conditions, with which the choice of the agent is determined.

Theorem 4. *If pressure of the world is a1 and evaluation of this pressure by the agent is a2, and the condition is satisfied*

$$a1 = \neg a2 \vee a1,$$

then a realistic choice of the agent is determined by pressure of the outside world: $a3 = a1$.

From the theorem 4 follows, that the determined choice of the agent is always determined by pressure of the outside world.

5. Conclusion

An agent under the conditions of choice may realize his choice on multi-element scales forming partially-ordered sets. In certain cases these sets form Boolean lattices. A set of norms defined on a Boolean lattice gives rise to many-values Boolean logic of norms. This permits us to describe the normalized behavior of the agent. The model of the behavior of an agent with normalized behavior includes situations where the agent is capable of possessing complete or partial freedom of choice.

Acknowledgments

I would like to express my gratitude to Vladimir Lefebvre and Harold Baker for their inestimable assistance in the writing of this work.

Bibliography

1. Lefebvre V.A. *A Psychological Theory of Bipolarity and Reflexivity*. Lewinston: The Edwin Mellen Press, 1992.
2. Lefebvre V.A. *The Cosmic Subject*. Russian Academy of Sciences, Institute of Psychology Press, 1996, 166 p.
3. *Proceedings of the Workshop on Multi-Reflexive Models of Agent Behavior* (Draft), Los Alamos, New Mexico, 18–20 August, 1998.
4. Taran T. A. A Possibility of the Logic of Argumentation Use for Multi-Agent Systems // *Proceeding of the International Workshop "Distributed Artificial Intelligence and Multy-Agent Systems" DAIMAS'97*. – June 15–18, St. Petersburg, Russia. – 1997. P. 224–234.
5. Birkhoff G. *Lattice Theory*. Providence, Rhode Island, 1967.

MULTI-AGENT REASONING AND SOCIAL CHOICE IN SCIENTIFIC RESEARCH PROGRAMMES

Fernando Tohmé[1], Claudio Delrieux[2]

[1]*Depto. de Economía, Universidad Nacional del Sur, Bahía Blanca, ARGENTINA*
[2]*Depto de Ing. Eléctrica, Universidad Nacional del Sur, Bahía Blanca, ARGENTINA,* claudio@acm.org

Abstract

Scientific theories incorporate a heterogeneous set of knowledge, hierarchically organized to represent a strategic dimension. Thus, in a given scientific domain, different programmes or theories may arise, each supported by a different group of researchers. In this work we investigate the issue of theory selection, i.e., the dynamic process by which theories are embraced or abandoned. First, we present a KR&R model for a scientific reasoning agent, based on Lakatos' Scientific Research Programmes. *We give a formal characterization of prediction and explanation, and other patterns of scientific inference, like induction, abduction, and theory change within this framework. Then we present a model of theory selection as a process of social choice among these agents, and the corresponding emergence conditions for Nash equilibra and coalition-proof Nash equilibra.*

Keywords: *Multi-agent Knowledge Representation and Reasoning (KR&R), Models of Uncertainty, Theory of Science, Social Choice and Game Theory.*

1.Introduction

Scientific theories incorporate a heterogeneous set of tentative knowledge which is hierarchically organized in terms of the context where the theory is operational. This organization represents the strategical dimension that accounts for the possible benefits of the *use* of that given knowledge by means of certain inference patterns in what is usually called the *scientific method.* Accepting or rejecting tentative knowledge may raise or lower the success or failure odds of a given theory. In Lakatos' *Scientific Research Programmes* [4, 5], two or more programmes or theories progress in a competitive fashion to give adequate predictions and explanations about a given scientific domain. These theories are not static, *i.e.*, some adoption or rejection of 'periferic' hypotheses is tolerated, but the hard core that identifies the programme as such must remain static. Relative success of a programme against the others may eventually lead to it's definitive preeminence. Then there is a tacit 'theory selection' process performed by the scientific reasoning agents, by means of which they decide to embrace one of the possible programmes within a discipline. Our claim here is that this process can be rationally understood in terms of game theory, and that theory selection is mostly a social choice process. In this work we present a model of theory selection as a social choice process. The starting point is to characterize the different *kinds* of knowledge arising in scientific research. Then we give a formal characterization of prediction and explanation, and other patterns of scientific inference. Finally, we study the stability conditions of our game-theoretical model, in particular, the emergence conditions for Nash equilibra and coalition-proof Nash equilibra.

2.KR&R and Research Programmes

The language of classical logic is insufficiently expressive to represent the different knowledge elements that constitute a research programme. For that reason we will establish in this Section some common points between scientific reasoning and nonmonotonic reasoning. Strict, deductive knowledge is represented in a deductively closed first order language $\mathcal{L}$. Such language is extended to allow the representation of other kinds of knowledge. Sentences representing lawlike statements assume the form of defeasible (default) rules. For instance, the normal default rule $\frac{a(X):b(X)}{b(X)}$ expresses that *"the disposition to accept $a(X)$ is a reason to tentatively accept $b(X)$"*. Sentences representing *particular* defeasible knowledge (*i.e.*, tentative evidence) are expressed as indexed ground literals l_i, where the disposition to consider particular knowledge l, is provided by a criterion i (exact or uncertain information conjecture, statistical criteria, etc.).

We will describe a scientific theory $\mathcal{T}$ as composed by the union of statements pertaining to $\mathcal{K}$, the logic and mathematical knowledge, $\mathcal{P}$, the principles of the given discipline, (for instance, Maxwell's laws), $\mathcal{H}$ represents the set of explicative hypotheses, *i.e.*, lawlike statements that are inferred from $\mathcal{P}$ (and $\mathcal{K}$) but have empirical content, and therefore are of more pragmatic value, $\mathcal{G}$ expresses the set of accidental generalizations that arises from an abstraction process over the set of observations and particular examples of phenomena, E, the evidence, is a set of experimental data represented as particular knowledge (ground literals), and C, finally, expresses a set of auxiliary hypotheses, each applicable to a given particular state of affairs to avoid the falsation of the theory. Usually $\mathcal{K}$ represents universally accepted knowledge. Since $\mathcal{P}$ are definitions that can not be deduced from other knowledge, a research programme can only take the strategic decision of accept or reject members of $\mathcal{P}$. The generalizations in $\mathcal{G}$ may be unsound (*i.e.*, may have exceptions), and we represent them with normal default rules. Members of C are tentative ground literals *i.e.*, particular knowledge relative to a given criterion. Thus, C is represented as a set of indexed ground literals.

DEFINITION 1 *Given a context $\mathcal{K}$, $\mathcal{P}$ an* **Epistemic Structure** *$\mathcal{E}_{\mathcal{K},\mathcal{P}}$ is a knowledge structure $\mathcal{E}_{\mathcal{K},\mathcal{P}} \subseteq \langle \mathcal{H}, \mathcal{G}, E, C\rangle$, where $\mathcal{H}$ is a consistent set of general knowledge such that $\mathcal{K} \cup \mathcal{P} \vdash \mathcal{H}$, $\mathcal{G}$ is a finite set of default rules, E is a set of particular knowledge, and C is a set of auxiliary hypotheses represented as tentative knowledge of the form l_i. When its context is clearly denoted, we will refer to an epistemic structure as $\mathcal{E}$.*

DEFINITION 2 *Given an epistemic structure $\mathcal{E}_{\mathcal{K},\mathcal{P}}$ within context $\mathcal{K}$,$\mathcal{P}$, we define a* **Theory** *$\mathcal{T}$ as a pair $\mathcal{T} = \langle \mathcal{E}_{\mathcal{K},\mathcal{P}}, \prec\rangle$, where the binary relation $\prec$ is a partial order on elements of $\mathcal{E}$, named* **Epistemic Preference Relation**. *$\mathcal{T}$ has an element $T_\top$ such that $\forall \alpha \in \mathcal{T}.\alpha \prec T_\top$ (i.e., deductive knowledge or evidence), and an element $T_\perp$ such that $\forall \beta \in \mathcal{T}.T_\perp \prec \beta$. Then $\mathcal{T}$ is a lattice under $\prec$.*

Given a set of knowledge $\langle \mathcal{H}, \mathcal{G}, E, C\rangle$ available in certain discipline, a theory selects some statements from each set. These statements need not be mutually consistent, since it suffices that they are consistent with the context and the set of evidence. The subset of statements that constitutes a theory are assigned an *arbitrary* partial order. This order represents the strategic importance that a research programme assigns to every piece of knowledge, being dependent from the external circumstances where the programme is developing. In such a way, in a given discipline, with a given set of available knowledge, there may exist several theories, each supported by a given epistemic importance relation.

DEFINITION 3 *Given a theory $\mathcal{T} = \langle \mathcal{E}, \prec\rangle$ and a subset $T \subseteq \mathcal{E}$, the* **Epistemic Importance** *of T given $\mathcal{T}$ is defined as the set of lower bounds of T under $\prec$: $\{\alpha \in T \nexists \beta \in T.\beta \prec \alpha\}$. Given two subsets of a theory, T_1 and T_2, we will say that T_1 is epistemically more important than T_2 (denoted as $T_2 \prec T_1$ for simplicity) ifff every statement in T_1 is at least as important in $\mathcal{T}$ as every statement in T_2, but there exists at least one statement in T_1 that is strictly more important than every statement in T_2.*

DEFINITION 4 *Given a theory $\mathcal{T} = \langle \mathcal{E}, \prec\rangle$, a* **Linear Extension** *l of the binary relation $\prec$ is a relation that includes $\prec$ and that induces a linear order on $\mathcal{E}$. Given a theory $\mathcal{T} = \langle \mathcal{E}_{\mathcal{K},\mathcal{P}}, \prec\rangle$ and a linear extension l of $\prec$, a* **Maximally Consistent Subset** *(SMC) of $\mathcal{T}$ (with respect to context $\mathcal{K}$,$\mathcal{P}$) is a consistent (wrt $\mathcal{K}$ and $\mathcal{P}$) subset $\mathcal{E}^l$ of $\mathcal{E}$ that satisfies:*

- $\forall \alpha \in \mathcal{E}^l.\forall \beta \in (\mathcal{E}/\mathcal{E}^l).\beta \prec \alpha$,
- $\nexists \mathcal{E}'.\mathcal{E}^l \subset \mathcal{E}' \subseteq \mathcal{E}, (\mathcal{E}' \cup \mathcal{K}) \nvdash \perp$,

The set $\mathcal{E}_{\not{\mathcal{L}}}$, then, is the intersection of all the SMC induced under every linear extension of $\prec$. Finally, the set C_T of **Conclusions** *(predictions or explanations) is the deductive closure of $\mathcal{E}_{\not{\mathcal{L}}}$, i.e., $C_T = Th(\mathcal{E}_{\not{\mathcal{L}}})$.*

It is worth to note that there is a computational procedure to determine if a given sentence is among the set of conclusions of a theory (see [2]).

3. Multi-Agent Theory Selection

Theory selection is a decision process in which several items of information are weighted in order to make a choice among theories. The results in the previous Section can be regarded as a general decision making process, where the basic statement is in terms of a single agent facing the uncertainty generated by non-intentional sources of information. The next step, considered here, is to introduce other agents interacting with different preferences. The outcomes are determined by the choices of all the parties. In the case of theory selection, a multi-agent framework seems natural to represent the social aspects of epistemology. In order to formalize the notion of social adoption of theories we start from the following assumptions. There are several agents, each one with a preferential ordering on the possible theories. As was stated in the previous Section, each agent's ordering constitutes a partial ordering on the set of theories. The choice of the majority is the adopted establishment. To be in the establishment is better than to be out. And finally, the adoption of an establishment is not instantaneous, since the process allows agents to learn the others' preferences and to act accordingly. The formal framework in which all these assumptions can be embedded is the following:

DEFINITION 5 *The theory selection process can be represented as a game $\langle N, T, \{\prec_i\}_{i\in N}\rangle$, where N is the set of reasoning agents, $T = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ is the set of theories in an epistemic structure $\mathcal{E}_{\mathcal{K},\mathcal{P}}$, and $\prec_i$ is agent i's preference ordering on elements of $\prod_{i\in N} S$. A* **profile** *$t \in \prod_{i\in N} T$ is $t = (\mathcal{T}_{i_1}, \mathcal{T}_{i_2}, \ldots, t_{i_N})$ where each $\mathcal{T}_i$ is the theory chosen by i.*

DEFINITION 6 *The social choice of a theory is given by a correspondence $\rho : \prod_{i\in N} T \rightarrow T$ such that $\mathcal{T}^* \in \rho(\mathcal{T}_{i_1}, \ldots, \mathcal{T}_{i_N})$ iff $|\{t_i = t^*\}| \geq |\{\mathcal{T}_i \neq \mathcal{T}^*\}|$*

So far, this framework does not provide a rationale for the preferences on profiles $\{\prec_i\}_{i\in N}$, which have to be carefully distinguished from the individual preferences on theories $\{\prec_i^T\}_{i\in N}$. To illustrate with an example, we will find a strategy that an agent must follow to become part of the establishment. A way of doing this for an agent i, given the joint choices of the others $\mathcal{T}_{-i} = (\mathcal{T}_1, \ldots, \mathcal{T}_{i-1}, \mathcal{T}_{i+1}, \ldots, \mathcal{T}_{|N|})$, is to generate preferences $(\mathcal{T}_i, \mathcal{T}_{-i}) \preceq_i (\mathcal{T}_i^*, \mathcal{T}_{-i})$ if and only if $(|\{\mathcal{T}_j = \mathcal{T}_i^*\}_{j\neq i}| > |\{\mathcal{T}_j = \mathcal{T}_i\}_{j\neq i}|) \vee (|\{\mathcal{T}_j = \mathcal{T}_i^*\}_{j\neq i}| = |\{\mathcal{T}_j = \mathcal{T}_i\}_{j\neq i}| \wedge \mathcal{T}_i \preceq_i^T \mathcal{T}_i^*)$ The set of assumptions implies that agents act according to the beliefs they have about the others' preferences. These beliefs represent the conjectures the agents make about the structure of the game. Given all these elements, a possible outcome should be predicted. A condition on the outcome that is natural to ask is stability. More precisely that it should be an equilibrium [3]:

DEFINITION 7 *A profile $(\mathcal{T}_1, \ldots, \mathcal{T}_i, \ldots, \mathcal{T}_{|N|})$ is a (Nash)* **equilibrium** *iff for each i there is no other $\mathcal{T}_i'$ such that $(\mathcal{T}_1, \ldots, \mathcal{T}_i, \ldots, \mathcal{T}_{|N|}) \prec_i (\mathcal{T}_1, \ldots, \mathcal{T}_i', \ldots, \mathcal{T}_{|N|})$*

Then, the following result shows the existence of an equilibrium:

PROPERTY 1 *There exists a Nash equilibrium for $\langle N, T, \{\preceq_i\}_{i\in N}\rangle$. Moreover, each equilibrium $(\mathcal{T}_1, \ldots, \mathcal{T}_{|N|})$ is such that $\mathcal{T}_i = \mathcal{T}_j$ for all $i, j \in N$.*

EXAMPLE 1 *There are three agents I, II and III. Each one can choose one from a set of theories $T = \{t_1, t_2, t_3\}$. Each agent has a preferential ordering on theories:*

$$I: \; t_2 \prec_I^T t_3 \prec_I^T t_1$$
$$II: \; t_3 \prec_{II}^T t_1 \prec_{II}^T t_2$$
$$III: \; t_3 \prec_{III}^T t_2 \prec_{III}^T t_1$$

The preferential orderings on profiles are represented by means of a payoff function $\sigma : T^3 \rightarrow \Re$, such that if for an agent i and profiles t and t', $t \prec_i t'$ then $\sigma(t') = 1$ while $\sigma(t) = 0$. Therefore, the preferential orderings on profiles can be represented by payoff matrixes such that the choices of the agents are as follows: agent I chooses rows, agent II columns and III matrixes. So, for example, the second row in the third column of the first matrix represents the profile (t_2, t_3, t_1).

$$\text{Matrix 1 is} \begin{bmatrix} 1,1,1 & 1,0,1 & 1,0,1 \\ 0,0,1 & 0,1,0 & 0,0,0 \\ 0,1,1 & 0,0,0 & 0,0,0 \end{bmatrix},$$
$$\text{matrix 2 is} \begin{bmatrix} 1,0,0 & 0,1,0 & 0,0,0 \\ 0,0,0 & 1,1,1 & 0,0,1 \\ 0,0,0 & 0,1,1 & 1,0,0 \end{bmatrix}, \text{ and}$$
$$\text{matrix 3 is} \begin{bmatrix} 1,1,0 & 0,0,0 & 0,0,0 \\ 0,0,0 & 0,1,0 & 0,0,0 \\ 0,0,0 & 1,0,0 & 1,1,1 \end{bmatrix}.$$

This means that if for example the profile of choices is (t_2, t_3, t_1), the payoff is, for each agent, 0. It is easy to check that the only Nash equilibria are supported by the profiles where all the agents choose the same theory.

As this example shows, an outcome which is less desired by the majority (in the example, t_3) may be supported anyway. This is a quite undesirable result, but it is clear that the only way to improve upon such a outcome is via a joint decision since no individual agent has incentives to deviate individually. But a coalition, i.e. a subset of N may join forces to deviate jointly away from the inferior outcome. The notion of *Coalition-Proof Nash equilibrium* [1] has been suggested as a representation of profiles immune to individual and joint deviation. Formally:

DEFINITION 8 *A profile $(\mathcal{T}_1, \ldots, \mathcal{T}_i, \ldots, \mathcal{T}_{|N|})$ is a* **coalition-proof** *Nash equilibrium iff there is no subgroup $N' \subset N$ and a profile $\mathcal{T}_{N'} = (\mathcal{T}_i')_{i \in N'}$ such that $(\mathcal{T}_1, \ldots, \mathcal{T}_i, \ldots, \mathcal{T}_{|N|}) \prec_i (\mathcal{T}_1, \ldots, \mathcal{T}_{N'}, \ldots, \mathcal{T}_{|N|})$*

4. Theory Selection and the Evolution of a Programme

Within a given discipline, theories share a common ground of which their epistemic structures are subsets. Criteria for theory selection, within the evolution of a programme, are based on the epistemic importance relation. This selection process normally occurs when a negative result (a failure in a prediction or explanation) forces the programme to evolve in order to assimilate the new evidence. The following example shows some different alternatives that can arise when confronted with a negative result.

EXAMPLE 2 *Let a theory $\mathcal{T}$ be $\mathcal{T} = \langle\{a, \frac{a:b}{b}\}, \{\}\rangle$. This theory predicts b. If b is not experimentally observed, i.e., if there is certain evidence that $\neg b$, then at least three new theories can be constructed from $\mathcal{T}$:*

1. *The new theory is $\mathcal{T}_1 = \langle\{a, \neg b, \frac{a:b}{b}\}, \{a \prec \neg b, a \prec \frac{a:b}{b}\}\rangle$. Following $\mathcal{T}_1$, the prediction fails because a is not adequately justified, but $\frac{a:b}{b}$ can be safely maintained. Moreover, this state of affairs suggests that presupposition that $\neg a$, which must be corroborated (see below).*
2. *Here we have $\mathcal{T}_2 = \langle\{a, \neg b, \frac{a:b}{b}\}, \{a \prec \neg b, \frac{a:b}{b} \prec a\}\rangle$. In $\mathcal{T}_2$, the culprit is the lawlike statement $\frac{a:b}{b}$, which is rendered false from evidence a and $\neg b$ that can be safely maintained.*
3. *Other cases may exist where an auxiliary hypothesis c is proposed to protect the lawlike statement from falsation. In this cases the new theory is $\mathcal{T}_3 = \langle\{a, c, \neg b, \frac{a:b}{b}, \frac{a \wedge c:\neg b}{\neg b}\}, \{\}\rangle$. Following $\mathcal{T}_3$, the statement $\frac{a:b}{b}$ systematizes only a subset of the domain, but a more specific law $\frac{a \wedge c:\neg b}{\neg b}$ must exist in a way such that completes the systematization in particular situations in which c is observed.*

But now we are in the same conditions as in the example of the previous Section, and according to the coalition proof criterion, a reasoning agent should accept $\mathcal{T}_1$.

It is important to state that other patterns of inference that lead to the discovery of new knowledge, as is the case of abduction, induction or hypothetical reasoning can be also stated as cases of theory selection. The inference pattern, in general, is that in order to accomodate a new (abductive, hypothetical or inductive) piece of information, other pieces of information must be withdrawn, thus leading to two or more possible new theories among to choose. Space considerations do not allow to fully develop these examples in this version of the paper.

References

[1] B. Bernheim, B. Peleg, and M. Whinston. Coalition-Proof Nash Equilibria: I Concepts. *Journal of Economic Theory*, 42:1–12, 1987.

[2] Claudio Delrieux. Nonmonotonic Reasoning Under Uncertain Evidence. In Fausto Giunchiglia, editor, *Artificial Intelligence: Methodology, Systems and Applications*, pages 195–204. Springer, Lecture Notes in Artificial Intelligence 1480, 1998.

[3] D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, Cambridge, Massachussetts, 1991.

[4] Imre Lakatos. *Proofs and Refutations. The Logic of Mathematical Discovery.* Cambridge University Press, 1976.

[5] Imre Lakatos. *The Methodology of Scientific Research Programmes. Philosophical Papers Vol. I.* Cambridge University Press, 1978.

DECISION MAKING PROCESSES IN MULTI-AGENT SYSTEMS

Nikolay G. Zagoruiko[1], **Yuri I. Zhuravlev**[2]

[1]*Institute of Mathematics SD RAS, pr. Koptiug, 4, Novosibirsk, 630090, Russia. E-mail: zag@math.nsc.ru*
[2]*Computer Centre RAS, Vavilov st. 40, 117968, Moscow, Russia. E-mail: zhur@ccas.ru*

Abstract

The tasks of monitoring the information related to work of multi-agent system are considered allowing discovering the regularities of structural organization of agents set and set of problems, solved by them. The new method of the collective decision making under the individual and group decisions with use of dispersion criterion of agents groups competence is offered.

Keywords: *monitoring, taxonomy, feature selection, decision making, prognosis, dispersion criterion, collective-group decision rules.*

1. Introduction

Let's imagine multi-agent system of decision making as a set *A* of the *M* agents *ai: a1, a2, ..., ai, ..., aM.* The sequence Z of tasks *z1, z2..., zj, ..., zN* is showed to the agents. The result *rij* of *ai*-th agent decision for a task *zj* is added to the protocol. The results of work of all multi-agent system can be represented by the table *R* of the decisions, consisting from *M* lines and *N* columns. *I*-th line of the table contains the decision results for the presented tasks by the *i*-th agent. Let's name a vector *ri1, ri2... rij... riN* as a "the profile of agent " *ai* in space of tasks *Z.* In *j*-th column of the table the decision results of *i*-th task by all agents are assembled. Let's name a vector *r1j, r2j... rij, ... rMj* as a "the profile of a task " *zj* in space of the agents *A.* Let's assume that each task *zj* is characterized by some set *L* "describing" properties *Y: y1j, y2j... ylj... yLj.* These properties can speak about the difficulty of the task, to what applied area it corresponds, who has put it, what for it is supposed to use the decision etc. The data such as "task - property" can be imagined as the table *B* with *N* columns and *L* lines, the information *blj* about value of *l*-th property for *j*-th task contained on their crossing.

Similarly, each agent can be characterized by a set of *K* "describing" properties X *x1, x2... xk, ... xK.* If *i*-th agent is understood as *i*-th set of decision rules, the properties *X* include, for example, number of rules in set, form of organization, specialization, level of effectiveness etc. This data also can be reduced in the table *D* "agent - property" by the *M* lines on *K* columns, where crossing contains the information *dki* about *k*-th property of *i*-th agent.

In result all available information on the agents and tasks may be represented by three adjacent tables *R, B* and *D.* Using each of them separately and all their combinations it is possible to put and to solve a wide range of monitoring tasks about a status of multi-agent system by methods of the data analysis [1]. Let's begin from the problem of the structure analysis of agents set and set of tasks by the using the taxonomy methods.

2. Taxonomy of the agents

If the taxonomy *Sar* of the agents according to "similarity" of their structures in space of tasks is made in the table *R*, the groups ("coalition") of the agents approximately equally deciding put tasks will come to light. It can give information about their identical abilities or identical positions in relation to arising problems.

It is possible to make taxonomy *Sad* of the agents on the data of table *D*. The agents with similar describing properties will be united into one taxon (coalition). It is of interest to compare the taxonomies *Sar* and *Sad*. If they coincide, it means, that there is a natural connection between values of describing properties *Y* of the agents and results *R* of the tasks decision by these agents. In every taxon it is possible to reveal the typical representative (precedent). If the resources of agent, making final decision ("supervisor") for observation over all agents are not sufficient, it is possible to limit supervision only by observation of precedents. Looking how they solve some task, it is possible to receive information about results of its solution by all other agents.

Looking how precedents solve some task, it is possible to predict the results of its solution by all other agents. The presence of connection between taxonomies allows recognizing to what coalition the new agent will enter according to its properties *X*.

3. Collective-group decision making

We described above the tasks of the data analysis reflecting the last results the decision of tasks *Z* by the agents *A*. The tables *R, B* and *D* represent a training material. Now we will address from the analysis of the last activity of the agents to the process of their current work on making of the cooperative decisions. Every *i*-th agent solves a task *z* and gives out his variant of the decision *ri*. Supervisor (agent accepting the final decision) makes the collective decision based on these individual variants. In that case it can start from the assumption about presence of some organized coalitions of the agents. Let the set of the agents is divided on *Q* independent groups, which obtain group decisions *rq*. The collective decision r is determined by the weighed averaging of the group decisions:

$$r = (\sum_{q=1}^{Q} rq * Jq) / \sum_{q=1}^{Q} Jq$$

How to determine competence of groups *Jq*? This question is important for all algorithms of pattern recognition using more than one decisive rules. The theoretical generalization of this approach within the framework of the algebraic theory of pattern recognition is presented in [2].

On the basis of the hypothesis of compactness [3] we formed the assumption that the decisions of the competent group of agents will differ from the decisions of it less competent competitors by a value of dispersion of the individual decisions received from the agents. The higher is the dispersion (entropy) of the group decisions, the higher will be the expected mistake of recognition or forecasting.

The check a informativness of the dispersion criterion was made within the framework of algorithm ZET for filling of gaps in the empirical tables and in the forecasting of the multi-dimension dynamic processes by means of the algorithm GAP [1]. The correlation coefficient between dispersion and mistake achieves the value of +0,7.

On this basis it is possible to offer the following general scheme of a class of effective algorithms for making the decisions by means of Collective - Group Decisive Rules (class of algorithms KGDR):

1) formation or detection agent groups,
2) making of the individual decisions and ratings of groups competence,
3) formation of the generalized decision and
4) rating of an expected mistake.

Bibliography:

1. Zagoruiko N.G. Applied methods of data and knowledge analysis. Ed. By Institute of Mathematics SN RAS, Novosibirsk, 1999. p. 213. (In Russian).
2. Yuri I. Zhuravlev. An Algebraic Approach to Recognition or
Classification Problems, Pattern Recognition and Image Analysis. 8(10), pp.59-100 (1998).
3. Zagoruiko N.G. Base Hypothesis in Methods of Data Analysis.// Pattern recognition and Image Analysis. N8 (v.4).
Interperiodica Publish, Moscow, 1998. pp.473-481

================================

The work was performed under support of Russian Fund for Fundamental Research (grant No.99-01-00582).

INDEX OF AUTHORS

Proceedings of the First International Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS`99)

Труды Первого Международного семинара Центральной и Восточной Европы по многоагентным системам

ЛП№ 000067

20.01.1999 г.

Подписано в печать 14.05.1999. Формат 60х90$^1/_8$.
Печ. л. 44,25. Печать офсетная. Заказ № 41. Тираж 250.
Издательство «Анатолия»
199178, Санкт-Петербург, В.О., 14-я линия, 39, СПИИРАН